Pflichtenheft

im Rahmen der Projektgruppe
„Generierung von Webanwendungen aus visuellen Spezifikationen“

Paderborn, den **29.08.04**

Inhaltsverzeichnis

1 Vorwort ... 3

2 Zielbestimmung ... 4

2.1 Aufgabe ... 4

2.2 Verwendete Werkzeuge ... 4

2.3 Vorgehen ... 6

3 Problembereich ... 7

3.1 Bespielszenario ... 8

3.2 Modell der Sprache ... 20

4 Anforderungsdefinitionen ... 23

4.1 Visuelle Sprache ... 23

4.2 Browser Navigations Knopf Problematik ... 32

4.3 Die Lesezeichen Problematik ... 33

4.4 Generierte Webanwendung ... 34

5 Technische Anforderungen ... 35

6 Glossar ... 36

1 Vorwort

Dieses Pflichtenheft ist Bestandteil der Entwicklungen im Rahmen der Projektgruppe „Generierung von Webanwendungen aus visuellen Spezifikationen“ an der Universität Paderborn, in der Arbeitsgruppe „Programmiersprachen und Übersetzer“ von Prof. Dr. Uwe Kastens.
Ziel des Projektes ist die Entwicklung einer visuellen Entwicklungsumgebung, mit dessen Funktionalität der Benutzer die Möglichkeit hat, durch Eingabe einer auf visuellen Konzepten basierenden Spezifikationssprache, eine vollfunktionsfähige und konsistente Webanwendung zu generieren.

2 Zielbestimmung

2.1 Aufgabe

In diesem Zusammenhang soll eine visuelle Entwicklungsumgebung mit Hilfe von DEViL generiert werden. Mit dieser ist der Benutzer in der Lage, eine vollständige Webanwendung basierend auf einem zustandsbasierten Modell visuell zu erstellen. Aus dem grafisch-visuellem Modell kann daraus der Programmcode der Webanwendung automatisch generiert werden.
Ein Hauptbestandteil der Aufgabe ist der Entwurf einer visuellen Spezifikationssprache für die Modellierung von Webanwendungen. Die visuelle Spezifikationssprache soll an UML-Statecharts angelehnt sein, wobei Zustände die Bildschirmmasken repräsentieren. Die Übergänge (Links) zwischen diesen sind als Verarbeitungsschritte zu interpretieren.
Zielsprache soll HTML mit Skript-Anteilen bestehend aus PHP-Anweisungen sein. Um Webanwendungen mit langlebigen (persistenten) Daten realisieren zu können, besteht die Notwendigkeit in der Einbindung einer Datenbank (DB). Dem Benutzer müssen Sprachkonstrukte zur Verfügung gestellt werden, mit denen er einfach und schnell Datenbankabfragen modellieren kann. Zu diesem Zweck ist zusätzlich die Datenbanksprache SQL in die Codegenerierung der Webanwendung zu integrieren.
Die visuelle Umgebung soll eine vollfunktionsfähige und konsistente Anwendung inklusive der bei Bedarf notwendigen Datenbankabfragen erzeugen. Dabei sollen Designaspekte aus der Zielanwendung nach Möglichkeit ausgegliedert werden, um das System wartbar und flexibel zu gestalten. Die Ergänzung der Anwendung um eigene Funktionen durch den Benutzer soll ermöglicht werden.

2.2 Verwendete Werkzeuge

Als Grundlage der geplanten Anwendung dient der an der Universität Paderborn entwickelte Generator DEViL. Der Generator erzeugt visuelle Entwicklungsumgebungen aus Spezifikationen der Sprachstruktur und der gra-

fischen Darstellung einer visuellen Sprache. Dabei wird die Verarbeitung (Transformation) der generierten Umgebung (ausgehend vom abstrakten Strukturbaum) in eine Zielsprache separat spezifiziert.
In Abbildung 1 wird dieser Sachverhalt grafisch verdeutlicht:

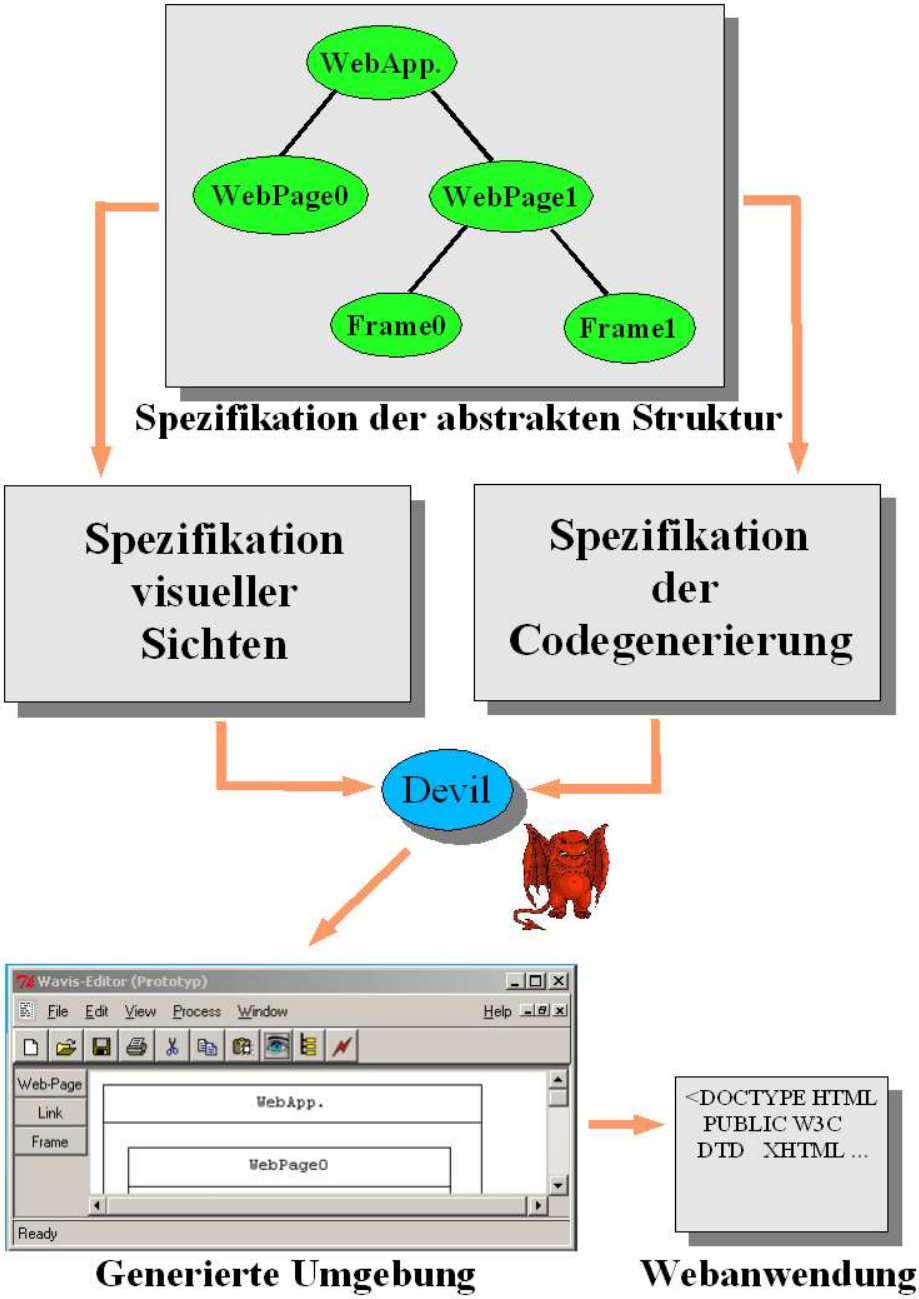


Abbildung 1 Generierung der visuellen Entwicklungsumgebung

Zu Beginn wird das Sprachmodell der visuellen Sprache spezifiziert. Die Spezifikation der Sprache des Modells basiert auf dem Konzept von Klassenhier-

archien. Klassen des Sprachmodells enthalten Attributdefinitionen und könne Eigenschaften von allgemeineren Klassen erben.
Aufbauend auf dem Sprachmodell werden die visuellen Sichten und die Codegenierung spezifiziert. Die Sichten legen visuelle Repräsentationen der abstrakten Struktur fest und können vom Benutzer dazu genutzt werden, eine Instanz des Sprachmodells, die Programmstruktur einer Webanwendung, zu erstellen. Mit Hilfe der definierten Code-Muster aus der Spezifikation der Codegenerierung wird die Programmstruktur einer Webanwendung dann in die anwendungsspezifische Zielsprache, zum Beispiel HTML mit eingebetteten Skriptsprachen-Anteilen (PHP), transformiert.

2.3 Vorgehen

Das geplante Ergebnis ist in Gruppenarbeit in einem zweistufigen Entwicklungsplan zu erzielen. Hierbei wird nach der Hälfe des Arbeitszeitraumes ein Prototyp zur Verfügung stehen, der einige der Grundfunktionalitäten enthält. In einer zweiten Phase wird die Arbeit verfeinert, um so alle geplanten Konzepte in das Produkt zu integrieren.

3 Problembereich

Im Folgenden wird ein Szenario vorgestellt, das später mit der zu entwickelnden visuellen Entwicklungsumgebung einfach und schnell, ohne PHP/HTML-Kenntnisse seitens des Benutzers, erstellt werden kann.

Beispielszenario: Webanwendung „PS-Autohaus“

Diese Webanwendung präsentiert ein Autohaus „PS-Autohaus“, das in einer Web-Umgebung verschiedene Autos dem Web-Anwender zum Kauf anbieten möchte. Als erstes wird eine SQL-Datenbank erstellt, in der die Datensätze der vorhandenen Autos importiert werden. Es existiert eine einzige Tabelle in der Datenbank, welche die Datensätze aller Autos enthält. Die Tabelle beinhaltet verschiedene Attribute wie Modell, Anzahl PS, Baujahr usw., die jedes Auto beschreiben. In einem zweiten Schritt wird die gesamte Webanwendung in zwei Sichten aufgeteilt, in Administrator- und Benutzersicht.
In der Benutzersicht existieren sieben Webseiten: „Home“, „Suche“, „Autoübersicht“, „Finanzierung“, „Kontakt“, „Mitarbeiter“ und „Hilfe“. Diese Seiten sind im Banner netzwerkartig untereinander verlinkt. Die Home-Seite enthält eine Begrüßung für die Benutzer.
Die Benutzer, die sich für ein bestimmtes Auto interessieren, können gezielt auf der Suchseite danach suchen, indem sie aus einer DropDown–Liste eine Automarke auswählen und den Button „Suchen“ anklicken, sofort werden ihnen alle vorhandenen Autos von der ausgewählten Automarke mit allen ihren Eigenschaften angezeigt.
Wenn der Benutzer sich alle in der Webanwendung angebotenen Autos anschauen möchte, so muss er auf die Seite „Autoübersicht“ gehen. Hier wird eine Tabelle mit allen in der Datenbank vorhandenen Auto-Datensätzen ausgegeben.
Nächste Seite ist die „Fananzierung“, hier kann der Benutzer überprüfen, ob er in der Lage ist ein bestimmtes Auto mit seinem Monatseinkommen und bestimmter Anzahl der Monatsraten zu finanzieren.
Die Kontaktseite enthält eMail-Adressen der Mitarbeiter des „PS-Autohauses“.

Aus der Hilfeseite befindet sich eine kurze Beschreibung zur Benutzung der ganzen Webanwendung.
Um von der Benutzersicht aus die Adminsicht zu gelangen, ruft man die Mitarbeiterseite auf. Hier muss er sein Login und Passwort eingeben. Nach einer erfolgreichen Anmeldung befindet sich der Benutzer, zum Beispiel ein Mitarbeiter, in der Adminsicht. Fast alle Funktionen in der Adminsicht sind mit denen in der Benutzersicht identisch. Zusätzlich kommen weitere Funktionen hinzu. Auf jeder Seite befindet sich ein Ausloggmechanismus, mit dessen Hilfe der Benutzer jederzeit die Adminsicht verlassen kann. Die zweite hinzugekommene Funktion ist auf der Autoübersichtseite zu sehen. Der Administrator kann durch das Anklicken des Radioknopfes ein bestimmtes Auto auswählen und editieren, d.h. die Eigenschaften verändern oder neue Bilder hinzufügen oder den ganzen Datensatz löschen. Alle vorgenommenen Veränderungen werden direkt in der Datenbank aktualisiert und gespeichert. Es kann auch jederzeit ein neuer Datensatz für ein Auto der Datenbank hinzugefügt werden.

3.1 Bespielszenario

Die größte Schwierigkeit, mit die der Entwickler einer Anwendung dieser Art konfrontiert wird, ist, dass PHP und SQL Kenntnisse vorhanden sein müssen. Durch das Einsetzen einer visuellen Entwicklungsumgebung zur Erstellung einer Webanwendung wird dem Benutzer diese Hürde abgenommen. So müssen nur die hierfür notwendigen funktionsspezifischen Sprach-Konstukte in der visuellen Umgebung ausgewählt werden und diese dann an der gewünschten Stelle platziert werden. Dabei muss der Benutzer nicht wissen, wie die Anmeldefunktion in PHP zu implementieren ist, sondern hat eine Funktion zur Verfügung, die diese Funktionalität bereitstellt, welche über eine Schaltfläche eingefügt werden kann.

Das hier vorgestellte Use-Case-Diagramm (Abbildung 2) spiegelt die Webanwendung Autohaus wieder, die aus sieben untereinander verknüpften Webseiten besteht und dem Betrachter eine Übersicht an Autos bietet.

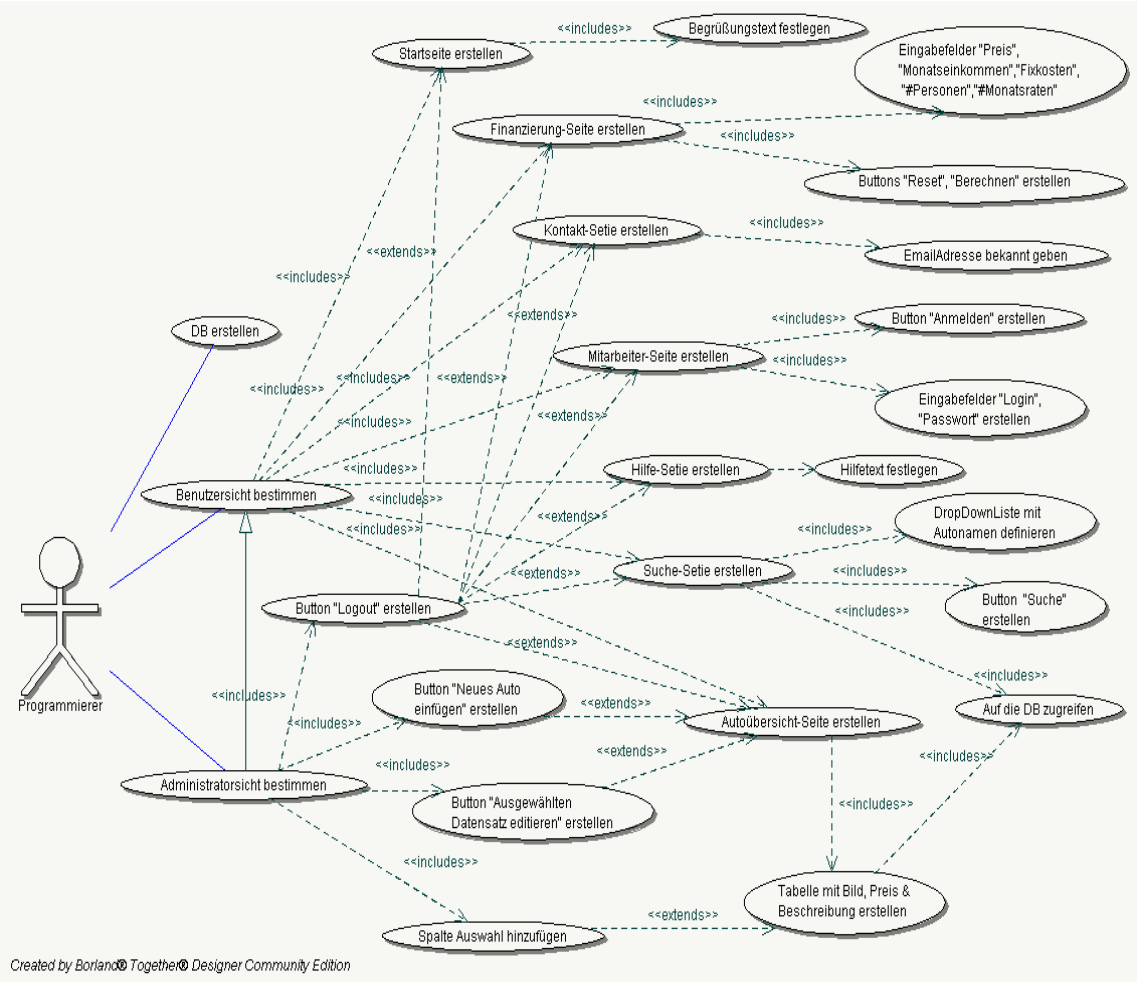


Abbildung 2 Use-Case-Diagramm: Autohaus

Die Webanwendung Autohaus bedient sich einer Datenbank, in der alle Autos mit dem Namen, dem Preis und der Beschreibung in einer Tabelle gespeichert sind. Auf den Seiten Autoübersicht und Suche der Autohaus-Webanwendung wird der Datenbankzugriff benötigt. Falls der Webanwendungsentwickler in seiner Webanwendung ebenfalls eine Datenbank einsetzen möchte, muss er den Use-Case 'DB erstellen' in der visuellen Umgebung auswählen. Diese Möglichkeit wird im Diagramm durch die Assoziation vom Benutzer zum Use-Case 'DB erstellen' verdeutlicht.

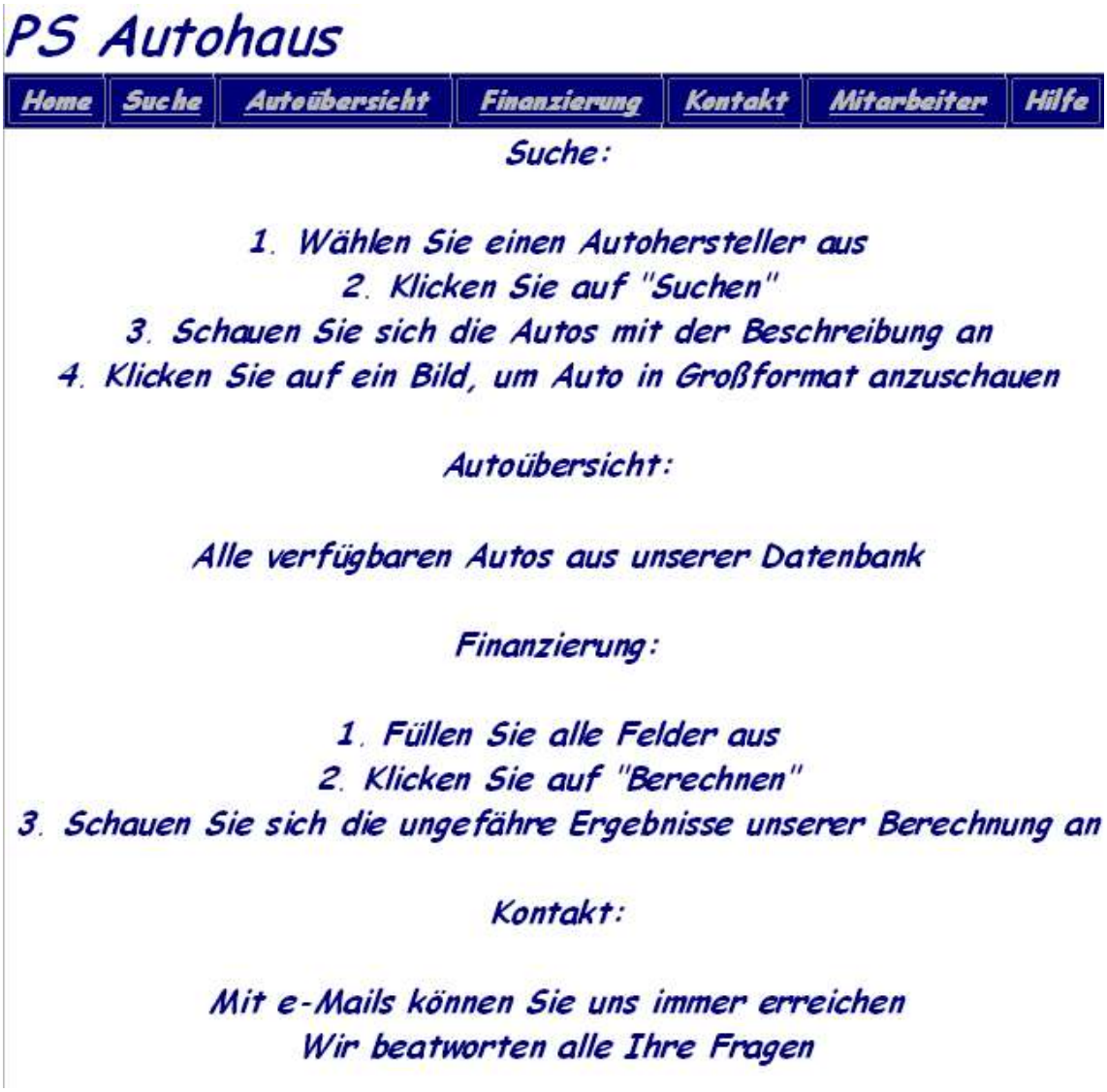
In dieser Webanwendung wurden zwei unterschiedliche Rollen realisiert. Den Bestand des Autohauses kann man sich in der Rolle eines einfachen Benutzers oder eines Administrators anschauen. Um diesen Sachverhalt zu verdeutlichen

wurden im Diagramm zwei Sichten als Use-Cases 'Benutzersicht bestimmen' und 'Administratorsicht bestimmen' definiert. Möchte ein Anwender solch eine dem Autohaus ähnliche Webanwendung erstellen, muss er mit der Bestimmung der Benutzer- und Administratorsicht beginnen.

Schauen wir uns erst einmal die Benutzersicht an. Da man sich als Anwender alle sieben Seiten der Webanwendung ansehen kann, gibt es sieben Use-Cases, je einen für die Erstellung einer Seite, die von dem Use-Case 'Benutzersicht erstellen' inkludiert werden. Dadurch, dass jeder Use-Case weitere Use-Cases importiert, verzweigt die Struktur etwas. Die Namen der Use-Cases sind hoffentlich selbsterklärend und geben ungefähr das Aussehen jeder Webseite wieder.
So würde man auf der Startseite einen vom Anwender festgelegten Begrüßungstext wiederfinden,

auf der Kontaktseite seine Emailadresse

und auf der Hilfeseite den von ihm verfaßten Hilfetext.



Auf der Autoübersichtseite bietet sich dem Betrachter die Auflistung unterschiedlicher Autos, die in einer Tabelle mit dem Bild, dem Preis und der Beschreibung zusammengefasst sind, wobei diese Daten aus der vorhandenen vom Webanwendungsentwickler erstellten Datenbank kommen.

PS Autohaus

Home | Suche | Autoübersicht | Finanzierung | Kontakt | Mitarbeiter | Hilfe

Alle Autos auf einem Blick

Bild	Beschreibung	Preis
	BMW 5, 200 PS, , Baujahr: 2004, Hubraum: 5.0l V12	20400 EURO
	BMW 5, 283 PS, , Baujahr: 2002, Hubraum: 2.5t	22300 EURO
	Lamborghini Countach, 450 PS, Farbe: silber, Baujahr: 2001, Hubraum: 2.8	265500 EURO
	BMW M3, 250 PS, Farbe: silber, Baujahr: 2004, Hubraum: 2.5t	24400 EURO
	BMW M3, 300 PS, Farbe: silber, Baujahr: 2003, Hubraum: 2.8	26000 EURO
	Bugatti Veyron, 400 PS, Farbe: blau, Baujahr: 2002, Hubraum: 3.0	225500 EURO
	Bugatti Veyron, 510 PS, Farbe: rot, Baujahr: 2003, Hubraum: 3.4	260000 EURO

Dieser Zusammenhang ist im Use-Case-Diagramm durch die drei Use-Cases 'Autoübersicht-Seite erstellen', ' Tabelle mit Bild, Preis und Beschreibung erstellen' und 'Auf die DB zugreifen' wiedergegeben.

Der Zugriff auf die Datenbank ist auch auf der Sucheseite unentbehrlich, wenn der Anwender sich Autos bestimmter Marke anzeigen lassen möchte.

Dies sollen die Use-Cases 'Suche-Seite erstellen', 'DropDownListe mit Autonamen definieren', 'Button „Suche“ erstellen' und 'Auf die DB zugreifen' repräsentieren.

Auf der Finanzierungseite kann der Anwender prüfen, ob er sich das von ihm ausgewählte Auto leisten kann.

Um dies zu realisieren importiert der Use-Case 'Finanzierung-Seite' die zwei anderen.
Der Use-Case 'Eingabefelder „Preis“, „Monatseinkommen“, „Fixkosten“, „#Personen“, „#Monatsraten“ erstellen' sorgt dafür, dass auf der Web-Seite die fünf Eingabefelder erscheinen, die vom Anwender ausgefüllt werden müssen. Alle Werte können per Resetbutton zurückgesetzt werden, und per Berechnenbutton kann das Ergebnis angezeigt werden, im Diagramm dargestellt durch den Use-Case 'Buttons „Reset“, „Berechnen“ erstellen'.

Die Mitarbeiterseite ermöglicht dem Anwender einen tieferen Einblick in die Webanwendung, wenn er sich mit einem Login und Passwort ausweisen kann.

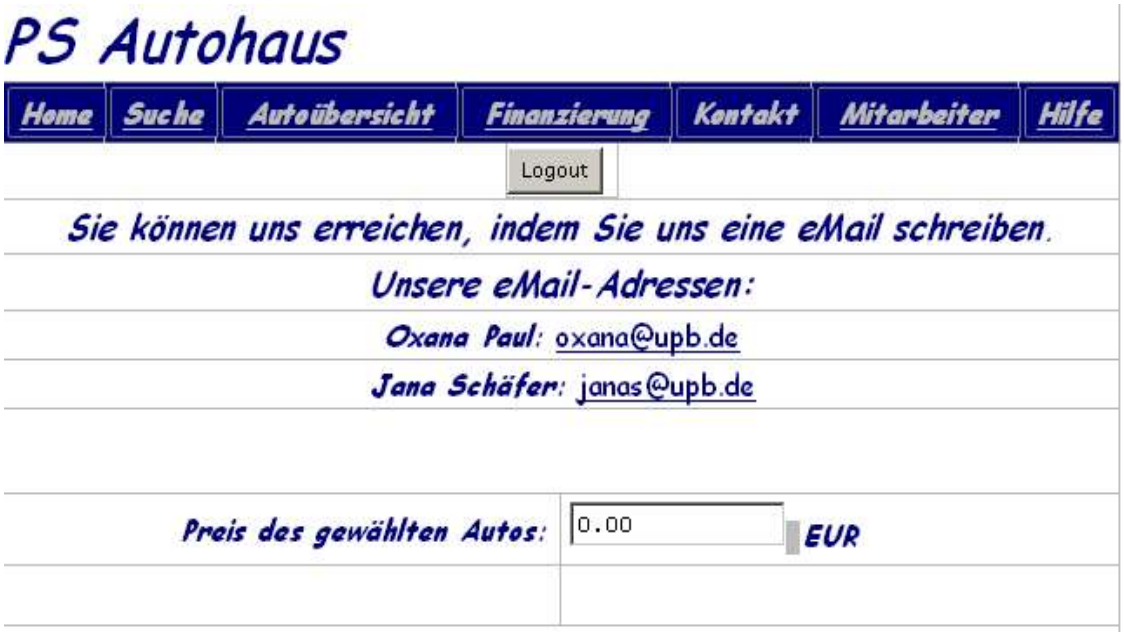
Mit den Use-Cases 'Mitarbeiter-Seite erstellen', 'Button „Anmelden“ erstellen' und 'Eingabefelder „Login“, „Passwort“ erstellen' wird der sogenannte Anmeldemechanismus modelliert. Hat man sich erfolgreich angemeldet, so hat man nun das Privileg in der Administratorsicht zu sein.

Die Administratorsicht ist sozusagen der Spezialfall der Benutzersicht, denn sie beinhaltet alles, was in der Benutzersicht vorhanden ist und enthält noch ein Paar Elemente mehr, die nur für den Administrator sichtbar sind. Diese Spezialisierung der Administratorsicht bzgl. der Benutzersicht äußert sich in der Vererbungsbeziehung, die durch einen nicht ausgefüllten Pfeil dargestellt wird. Das bedeutet, dass alle Use-Cases, die von der Benutzersicht importiert werden, auch von der Administratorsicht importiert und benutzt werden.
Nach der Authentifizierung erscheint auf jeder Webseite bis auf die Mitarbeiterseite ein zusätzlicher Button „Logout“.

PS Autohaus
Home
Suche
Autoübersicht
Finanzierung
Kontakt
Mitarbeiter
Hilfe
Logout
Wählen Sie den gewünschten Autohersteller
Suchen

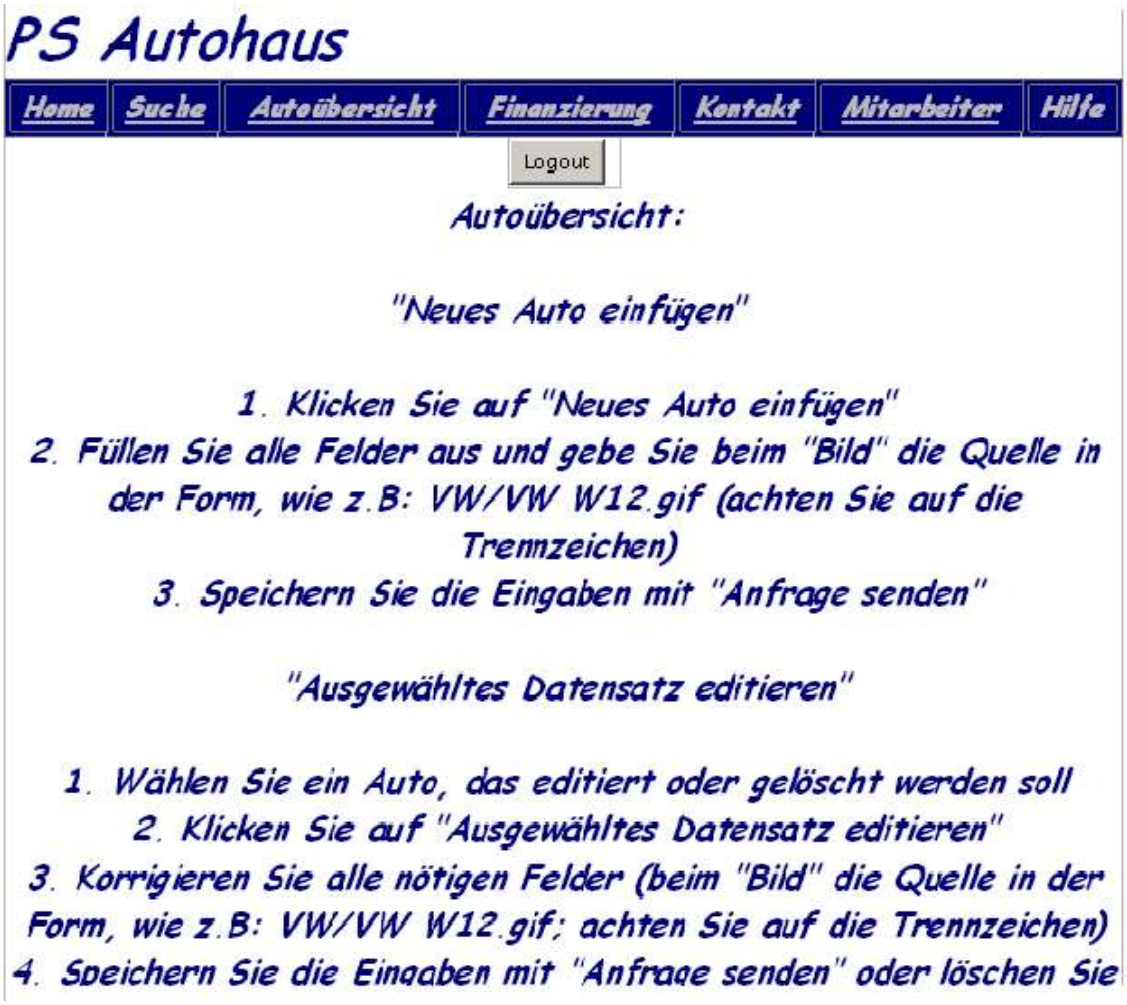
PS Autohaus
Home
Suche
Autoübersicht
Finanzierung
Kontakt
Mitarbeiter
Hilfe
Logout
Neues Auto einfügen
Ausgewählten Datensatz editieren
Auswahl
Bild
Beschreibung
Preis
<>
BMW 5, 200 PS, , Baujahr: 2004, Hubraum: 5.0l V12
20400 EURO
<>
BMW 5, 283 PS, , Baujahr: 2002, Hubraum: 2.5t
22300 EURO
<>
Lamborghini Countach, 450 PS, Farbe: silber, Baujahr: 2001, Hubraum: 2.8
265500 EURO
<>
BMW M3, 250 PS, Farbe: silber, Baujahr: 2004, Hubraum: 2.5t
24400 EURO
<>
BMW M3, 300 PS, Farbe: silber, Baujahr: 2003, Hubraum: 2.8
26000 EURO
<>
Bugatti Veyron, 400 PS, Farbe: blau, Baujahr: 2002, Hubraum: 3.0
225500 EURO

In dieser Sicht ist nun der Inhalt der Autoübersichtseite ein anderer. Es befinden sich hier zwei neue Buttons mit den Beschriftungen „Neues Auto einfügen“ und „Ausgewählten Datensatz editieren“, mit denen der Administrator den Autobestand und die abgespeicherten Informationen verändern kann. Die Tabelle bekommt in dieser Sicht eine neue Spalte „Auswahl“, die aus den Radiobuttons besteht und somit die Selektion einzelner Autos erlaubt. Dieser Sachverhalt wird im Diagramm durch die vier Use-Cases verdeutlicht, die von dem Use-Case 'Administratorsicht bestimmen' importiert werden. Der Use-Case 'Spalte Auswahl hinzufügen' erweitert über die <<extends>>-Beziehung den Use-Case 'Tabelle mit Bild, Preis und Beschreibung erstellen', indem es zum neuen Layout und Inhalt der Autoübersichtseite kommt. Die restlichen drei Use-Cases erweitern den Use-Case 'Autoübersicht-Seite erstellen' und bewirken, dass auf dieser Seite drei neue Schaltflächen entstehen.



Auf der Mitarbeiterseite ersetzt dieser Button alle Elemente von der früheren Sicht.

Das Anzeigen dieses Buttons wird im Diagramm durch den Use-Case 'Button „Logout“ erstellen' signalisiert, der vom Use-Case 'Administratorsicht bestimmen' importiert wird und alle anderen Use-Cases für die Webseiten erweitert, dargestellt durch die Pfeile mit der Beschriftung <<extends>>.



Das Use-Case-Diagramm in Abbildung 3 soll veranschaulichen, welche Eigenschaften die Webseite, die vom Anwender mit Hilfe der visuellen Umgebung erstellt wird, haben kann.

In dem Struktureditor der visuellen Umgebung legt der Anwender das Layout und die Mächtigkeit seiner Webseite fest, indem er die für ihn in Frage kommenden Use-Cases auswählt.
Handelt es sich nicht nur um eine einfache Informationsseite, sondern soll sie auch ein Formular enthalten, das später ausgewertet wird, so dürfen die Use-Cases 'Eingabedaten validieren' und 'HTML-Seiten mit Objekten füllen' bei der Erstellung einer solchen nicht fehlen.

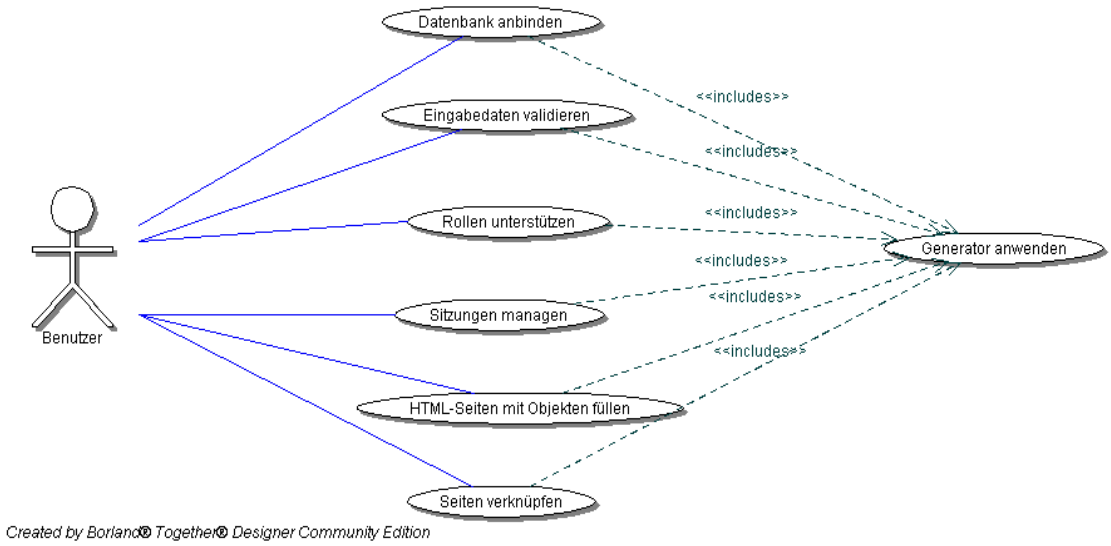


Abbildung 3 Use Case Diagramm: Funktionalitäten

In den meisten Fällen möchte man auch eine übersichtliche Navigationsstruktur haben um dem Besucher der Webseite eine gute Orientierung zu geben. Die Voraussetzung hierfür ist die Verknüpfung der Web-Unterseiten miteinander. Der Anwender würde diesen Wunsch an die visuelle Umgebung durch die Auswahl des Use-Cases 'Seiten verknüpfen' vermitteln.
Falls ein Sitzungsmanagement erwünscht ist, ist dafür der Use-Case 'Sitzungen managen' anzugeben. Damit können beliebig viele Sitzungen geöffnet, verwaltet und geschlossen werden, die Cookies können gesetzt werden.
Mit dem Use-Case 'Rollen unterstützen' soll dem Anwender die Möglichkeit eingeräumt werden verschiedene Rollen zu vergeben und je nach Rolle die Objektauswahl auf der Webseite zu gestalten. Für den ersten Prototypen wird nur die Unterscheidung in zwei Rollen stattfinden, die des Administrators und des einfachen Benutzers.
Hat der Anwender viele Daten strukuriert in einer Datenbank liegen und möchte er diese mit seiner Webseite in Verbindung setzen, so sollte dies durch den Use-Case 'Datenbank anbinden' erreichbar sein.
Jeder von den hier beschriebenen Use-Cases importiert einen weiteren Use-Case 'Generator anwenden', der die eingentliche Arbeit übernimmt und die gewünschte Webseite nach den vorher festgelegten Kriterien generiert.
Das Ziel ist hierbei, auch einen Anwender mit wenig oder keinen Programmierkenntnissen der Erstellung professioneller Webseiten zu bemächtigen.

3.2 Modell der Sprache

Das Klassendiagramm in Abbildung 4 beschreibt die Sicht der visuellen Umgebung bzw. des Struktureditors auf eine Webanwendung.

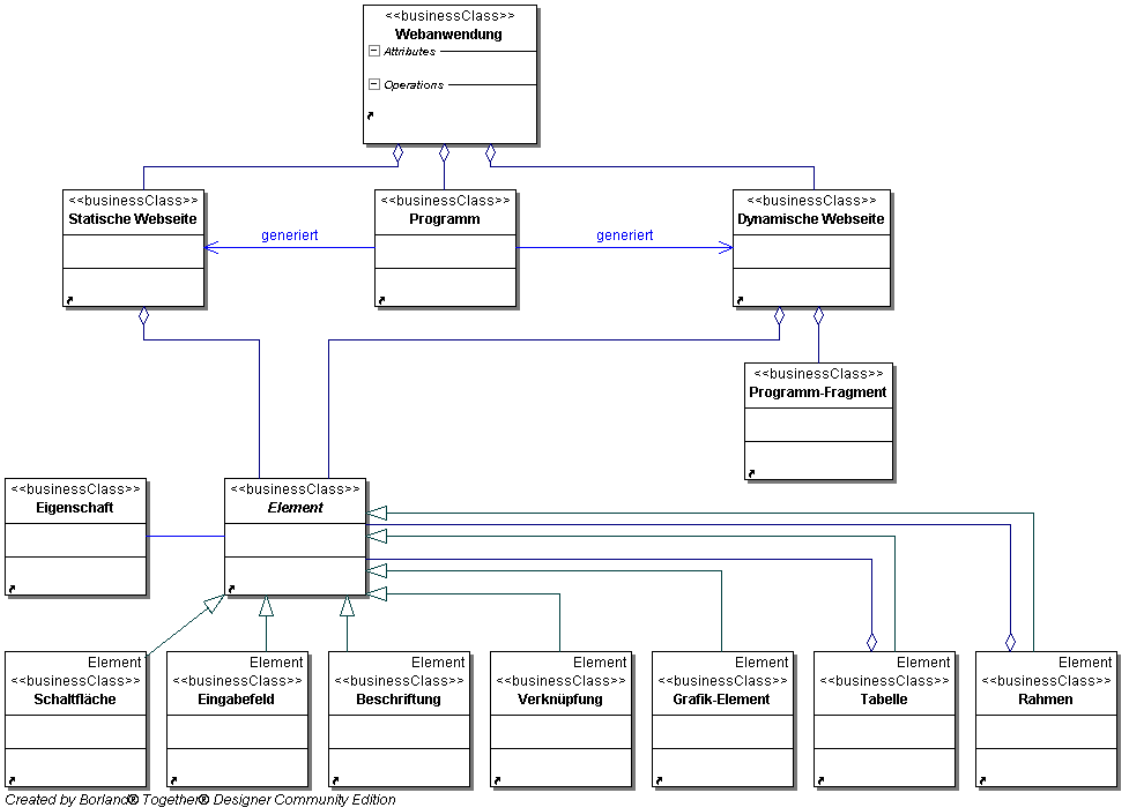


Abbildung 4 Klassen-Diagramm: Problembreich

Eine Webanwendung besteht aus statischen und dynamischen Webseiten, sowie aus Programmen. Unter Programme werden in diesem Fall eigenständige PHP-Programme verstanden, die statische oder dynamische Seiten als Ausgabe erzeugen können. Dynamische Seiten können weitere Programm-Fragmente enthalten, die beim Aufruf der Seite ausgeführt werden. Statische Seiten werden nach ihrer Erzeugung nicht mehr verändert.
Statische und dynamische Webseiten bestehen aus Elementen. Diese Elemente beschreiben den Aufbau einer Webseite. Es handelt sich dabei um Elemente, wie Schaltflächen (Buttons),
Eingabefelder für Text, Beschriftungen, Verknüpfungen mit anderen Webseiten, Grafik-Elemente (Grafiken, Animationen, etc.), Tabellen und Rahmen. Man

kann hier zwischen verschiedenen Arten von Elementen unterscheiden. Es gibt zum Einen reine Gestaltungs-Elemente, wie Rahmen, Grafiken, Beschriftungen und Formular-Elemente wie Schaltflächen oder Eingabefelder.
Rahmen sind das zentrale Design-Element einer Webseite, da sie eine Gruppierung und Ordnung der Inhalte der Seite ermöglichen. In ihnen können alle anderen Element-Typen enthalten sein. Es können insbesondere auch beliebig viele Rahmen ineinander verschachtelt werden, was komplexere Seiten-Designs ermöglicht. Bei diesen Rahmen handelt es sich nicht um HTML-Frames!
Tabellen dienen ausschließlich zur Darstellung von Daten. Sie sind, im Gegensatz zu Rahmen, nicht dazu gedacht, Elemente einer Seite zu gruppieren. Es ist jedoch auch möglich, dass eine Tabelle beliebige andere Elemente enthalten kann.
Jedem dieser Elemente ist eine bestimmte Anzahl von Eigenschaften zugeordnet. Diese Eigenschaften beschreiben zum Beispiel das Aussehen (Farbe, etc.) und das Verhalten der Elemente. Der Benutzer der visuellen Umgebung soll diese Eigenschaften durch die Verwendung von Formatvorlagen beeinflussen können.

Eine nach diesem Modell aufgebaute Webanwendung kann auf einer Architektur, wie sie im Deployment-Diagramm (Abbildung 5) dargestellt ist, ausgeführt werden. Die Webanwendung wird dabei auf einem Web-Server installiert, der an einen (optionalen) Datenbank-Server gekoppelt ist. Abfragen auf dieser Datenbank werden innerhalb der Webanwendung von Programmen bzw. Programm-Fragmenten durchgeführt. Bei diesen Programmen kann es sich um bereits vorgefertigte Programme (Passwort-Abfrage, etc.) oder um Programme, die der Entwickler der Webanwendung selbst anfertigt, handeln. Diese Programme werden während der Laufzeit der Anwendung durch den Web-Server ausgeführt.

Durch diese Modell-Beschreibung ist es nun möglich, einzelne Elemente der visuellen Sprache zu entwerfen, welche zur Erstellung einer Webanwendung verwendet werden können.

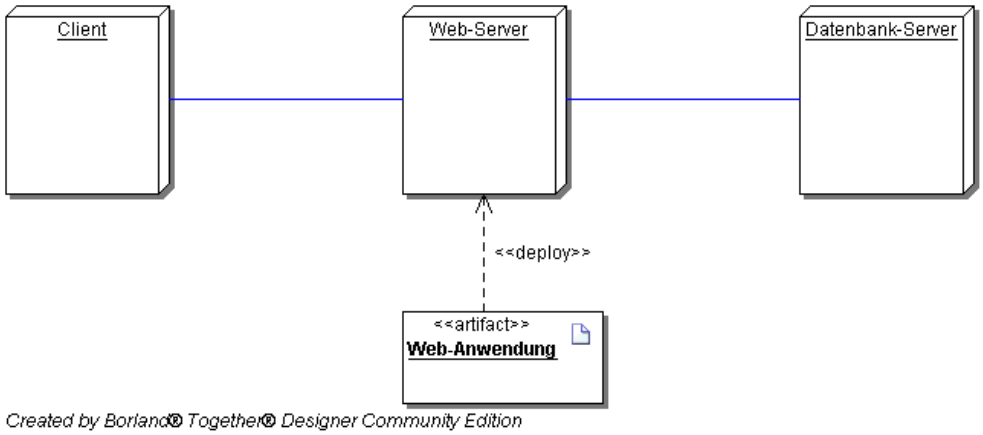


Abbildung 5 Problem Model Deployment

4 Anforderungsdefinitionen

In dieser Projektgruppe soll eine visuelle Entwicklungsumgebung entstehen, mit der man eine Webanwendung zunächst visuell modellieren kann und daraus dann eine vollfunktionsfähige Webanwendung auf Basis von HTML und PHP generieren lassen kann.
Die visuelle Umgebung soll in der Lage sein, zunächst nur einfache, im zweiten Prototyp auch komplexere, dynamische Webanwendungen zu erstellen, die als Zielsprache HTML und PHP benutzen. Zusätzlich wird eine Datenbank angebunden, zunächst soll nur MySQL unterstützt werden, später auch andere relationale Datenbanksysteme.
Aus dem späteren Einsatzbereich des Systems ergeben sich sofort Anforderungen an die visuelle Entwicklungsumgebung, die visuelle Sprache und die Webanwendung:

4.1 Visuelle Sprache

Die visuelle Sprache soll möglichst einfach und schlicht gehalten werden, damit der Webanwendungsentwickler der Webseite möglichst wenig Einarbeitungszeitbenötigt. Ein Webseitenprojekt soll dabei aus verschiedenen Webseiten bestehen, die in einer Navigationssicht (Root View) miteinander verknüpft werden können. Eine einzelne Seite besteht dann aus einer Reihe von Komponenten.

In Abbildung 6 folgt eine etwas genauere Betrachtung der visuellen Sprache:
Abbildung 6 Problem Model Deployment

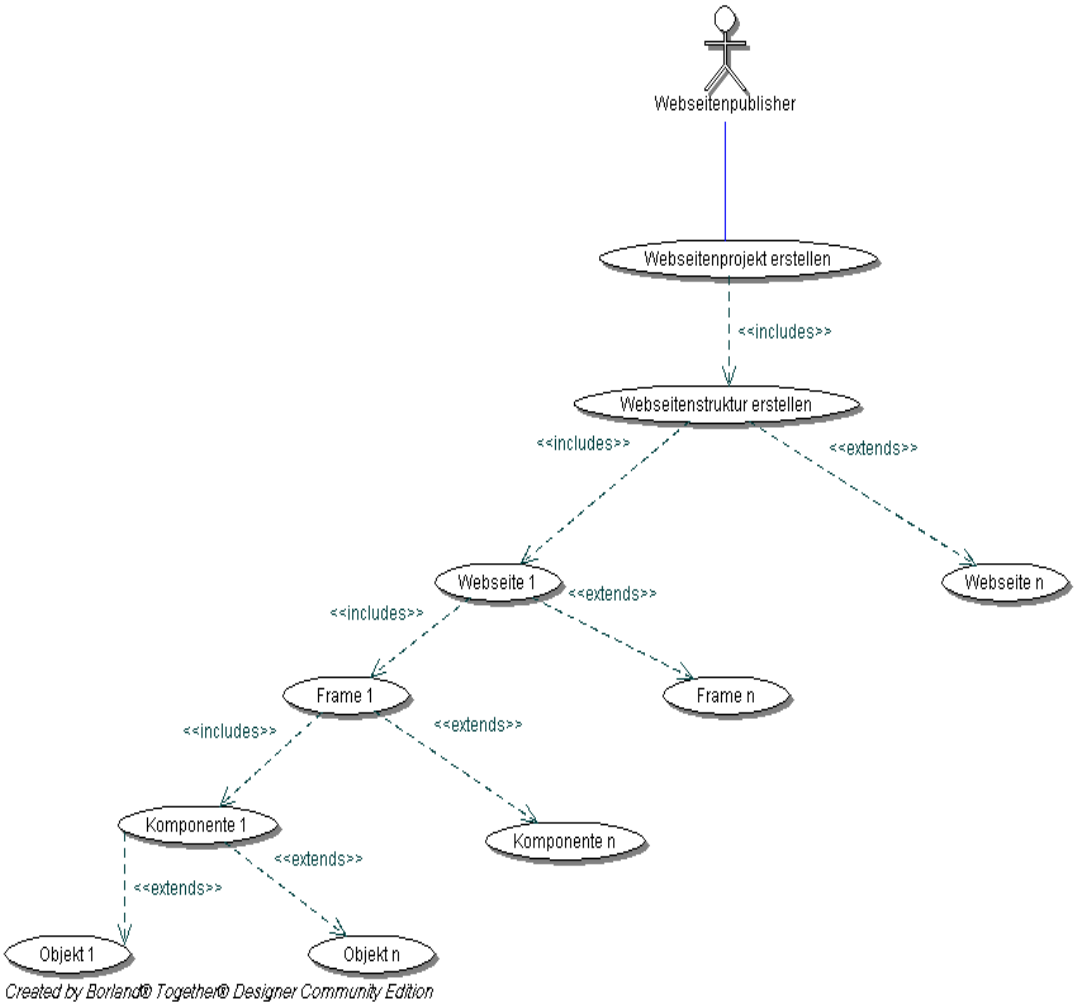


Abbildung 6 Die visuelle Sprache

Der Webanwendungsentwickler legt zunächst in der Root View die Seitenstruktur der zu erstellenden Webseiten fest. Hier wird die Webseiten-Struktur als eine Art 'Sitemap' dargestellt. Um die Webseiten-Struktur übersichtlich zu halten, werden nur die Seiten mit Titel und die Links dazwischen angezeigt. Der Webanwendungsentwickler hat (bisher) die Möglichkeit aus 3 Komponenten zu wählen: Die einzelnen Webseiten bestehen aus rechteckigen Kästen mit dem Titel der Webseite als einziges Attribut. Verbindungen (Links) zwischen Webseiten werden durch gerichtete Pfeile gekennzeichnet. Zusätzlich gibt es besonders gekennzeichnete Pfeile, die darauf hindeuten sollen, dass die Verfolgung des Links nur bestimmten Benutzerrollen vorbehalten ist.

Zusätzlich gibt es ein Piktogramm "Verteiler", dass eine Art Navigationselement darstellt. Zeigt eine Webseite auf diesen Verteiler, so wird von dieser Webseite auf alle Webseiten verlinkt, die mit dem Verteiler verbunden sind.
Klickt der Benutzer auf ein Webseitenelement, so erscheint die Webseitensicht. Hier wird auf eine einzelne Webseite gezoomt und deren Inhalt dargestellt.
Eine Webseite wird als Rechteck dargestellt und enthält mindestens einen Rahmen, welcher als Container für weitere Rahmen und/oder andere Komponenten dient. In diesen - nicht mit HTML-Frames zu verwechselnden - Rahmen können beliebig viele Komponenten gelegt werden. Komponenten sind dabei Formulare, Links, Texte und andere Mediaelemente.
Nun folgend eine Übersicht über die Elemente, wie sie der Editor zur Generierung einer Webseite anbietet. Diese sind in der Reigenfolge geordnet, wie sie ein Webanwendungsentwickler bei der Erstellung berücksichtigen sollte.

Element	Funktionalität
Website	Festlegung von Datenbankparametern, Ausgabeverzeichnissen sowie weiteren Projekteigenschaften.
Webseite	Legt grundlegende Eigenschaften der Webseite fest und besitzt Links zu anderen Webseiten. Zudem kann einer Webseite auch eine Stylesheet/Formatvorlage auferlegt werden. Eine Webseite enthält nichts anderes als Rahmen.
Link / Konnektor	Mit ihm wird eine Verbindung zu einer anderen Webseite oder Skript festgelegt. Links können sich jedoch auch auf bestimmte Benutzergruppen beschränken, dürfen also ggf. nicht von jedem Benutzer genutzt werden. Mit Konnektoren lassen sich übersichtliche Navigationsstrukturen für z.B. Navigationsleisten implementieren.
Rahmen	Rahmen beinhalten jegliche Mediaelemente sowie ggf. weitere Rahmen. Sie dienen z.B. zur Abgrenzung von Navigationsleiste und Inhalt. Ein Rahmen kann zudem auch Formulare enthalten.

Element	Funktionalität
Formular	Ein Formular ist eine Sammlung von Eingabefeldern, Buttons und Labels und entspricht einem inhaltlich abgeschlossenen Eingabedialog, welcher aus Gründen der Arbeitserleichterung nicht immer wieder neu erstellt werden muss.
Mediaelemente	Mediaelemente können Texte, Bilder, Sounds, Eingabe-elemente, Textboxen, Input-Fields, Buttons sowie alle sonstigen verfügbaren HTML-Elemente sein.

Ein Formular besteht aus einem Rechteck, in dem die zugehörigen Formularelemente liegen. Von Formularen gehen Pfeile an den Rand des Rahmens zu einem "Konnektor" Element. Dieser Konnektor kann wiederum auf ein externes Skript verweisen. Ein externes Skript wird als Rechteck dargestellt. In dieses Skript muss der Webanwendungsentwickler seinen PHP-Code von Hand eintragen.
Parameter, die das Skript benötigt, werden vom Formular automatisch übergeben. Von den Skript-Elementen führt genau ein Link auf ein anderes Skript oder auf eine andere Webseite.
Abbildung 7 zeigt das Zusammenspiel von Webanwendungsentwickler, Generator, Administrator und Webanwendungsnutzer.

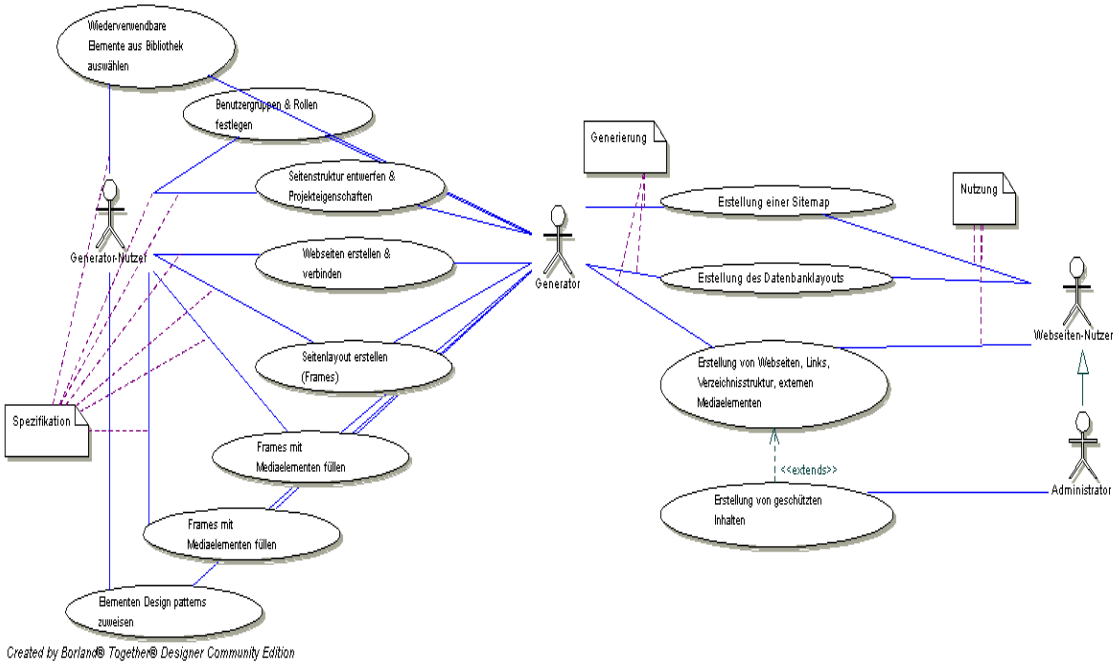


Abbildung 7 Zusammenspiel von Webanwendungsentwickler der Website, Generator, Administrator und Webanwendungsnutzer

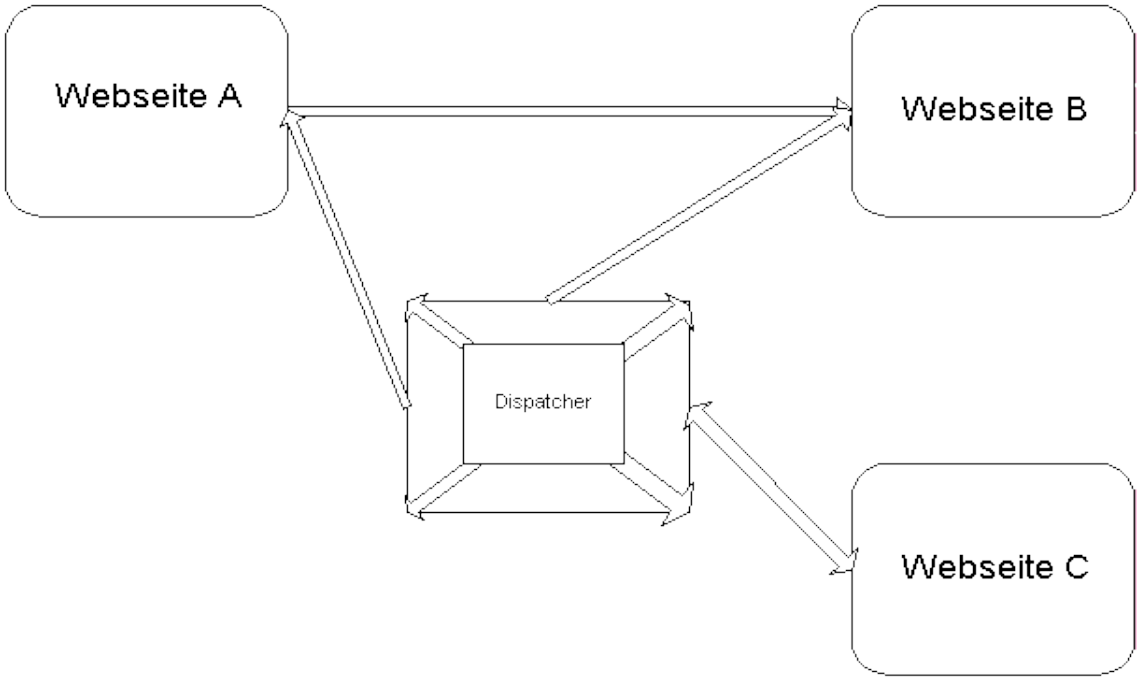


Abbildung 8 'Root View' mit Navigationsstruktur

In Abbildung 8 sieht man die 'Root View', in der die Navigationsstruktur festgelegt wird. Webseiten werden durch Rechtecke dargestellt, die durch gerichtete Pfeile miteinander verlinkt werden können.

Als Besonderheit exisitiert noch ein Dispatcher, der als Navigationsleiste fungiert. Wenn Webseiten auf diesen Dispatcher verlinken, dann werden Sie automatisch mit allen Webseiten verknüpft, auf die der Dispatcher zeigt.

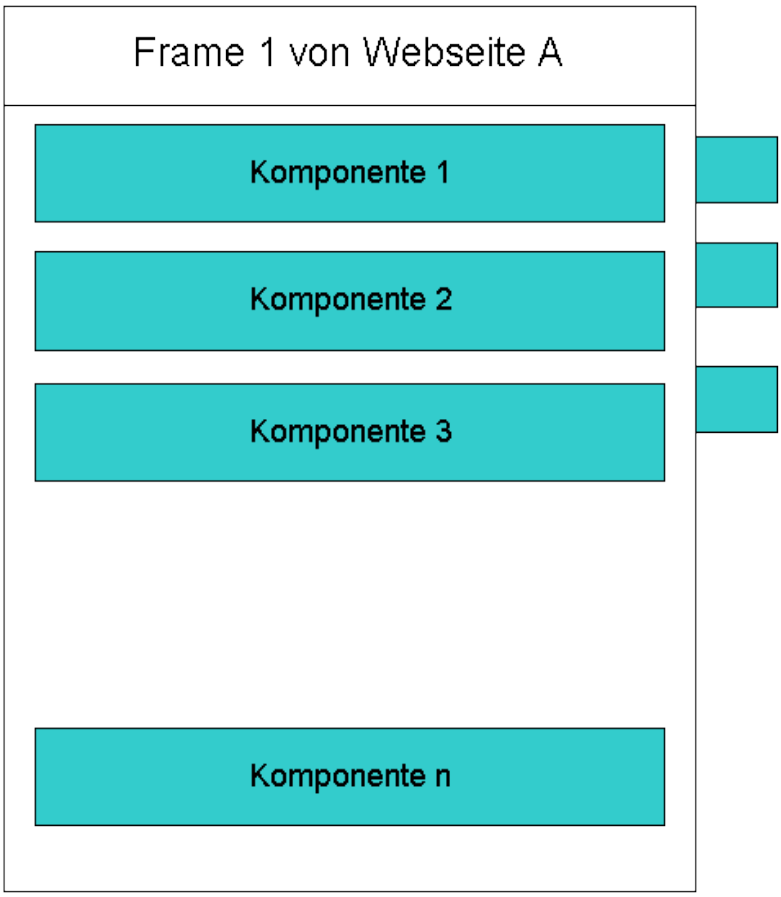


Abbildung 9 Realisierung einer Webseite

Eine Webseite besteht aus mindestens einem Rahmen. In Abbildung 8 ist ein solcher Rahmen zu sehen. Rahmen haben einen Namen und enthalten beliebig vielen Komponenten. Komponenten können sein: Textobjekte, Links, Mediaobjekte oder Formulare.
Am Rand der Rahmen sind Konnektoren zu sehen. Konnektoren werden mit anderen Webseiten oder Skriptelementen verbunden.
Ein Formular würde also zunächst mit einem Rahmen-Konnektor verbunden und dieser dann mit einem Skriptobjekt.

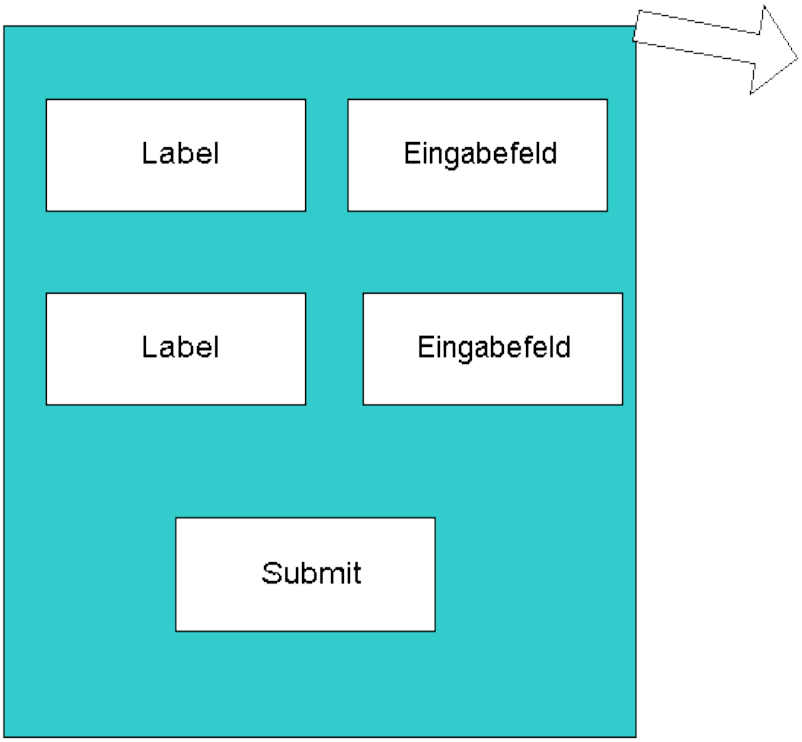


Abbildung 10 Beispiel: Login-Dialog

Abbildung 10 zeigt ein Formular, dass einen Login-Dialog darstellen könnte. Es besteht aus 2 Labeln, 2 Texteingabefeldern und einem Submit-Button. Der Pfeil zeigt auf einen Rahmen-Konnektor.

Abbildung 11 Import von Skriptanteilen

Ein Skriptelement enthält PHP-Code, der entweder vom Anwender eingegeben wird oder, falls das Skript mit der Datenbank angebunden wird, automatisch erstellter Code.

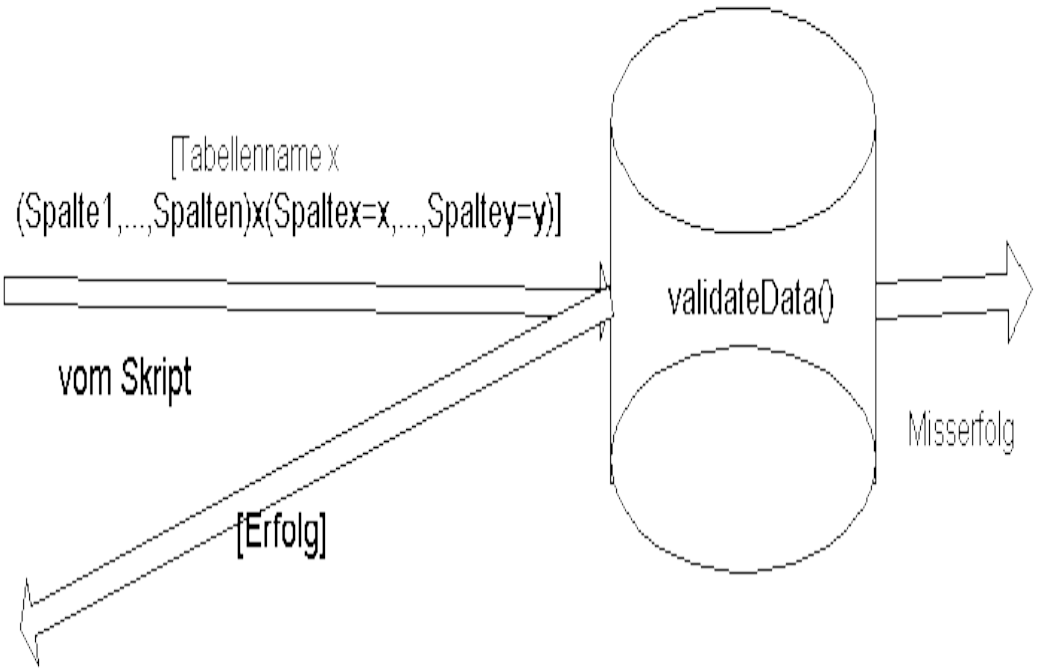


Abbildung 12 Nutzung von Standardfunktionalitäten

Abbildung 12 zeigt eine Methode, die eine Standardfunktionalität bereitstellt. In diesem Fall wird ein Datum validiert.
Dabei kommen folgende Informationen vom oben erwähnten Skriptelement:
[Tabellenname x (Spalte1,...,Spalten)x(Spaltex=x,...,Spaltey=y)]
Die validateData() Methode spricht die Datenbank dann mit einer SQL-Anfrage, wie select Spalte1,Spalten from Tabellenname where Spaltex=x and Spaltey=y, an. Es wird die Anzahl der Zeilen zurückgegeben.
Falls 0 zurückgegeben wurde, wird zum Skript zurückgeleitet, andernfalls zu einer anderen Webseite.

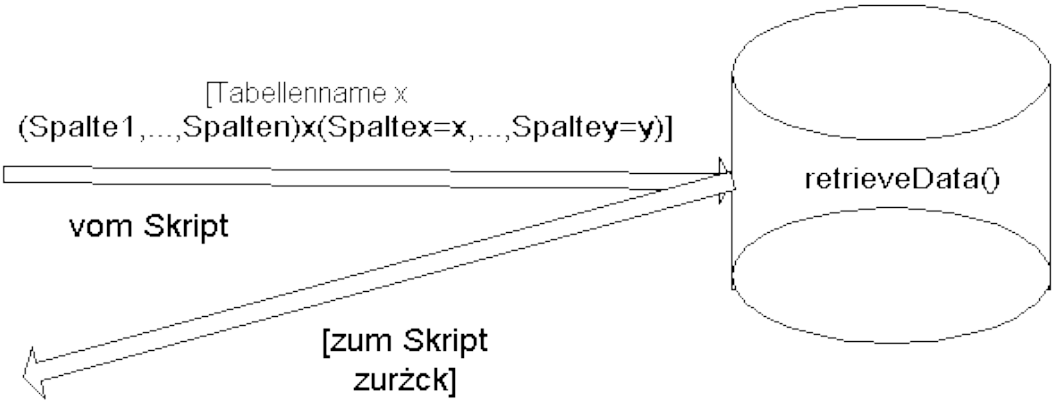


Abbildung 13 Abfragen von Datenbankinhalten

Die in Abbildung 13 dargestellte Methode sorgt dafür, dass eine beliebige Anzahl von Datenbankeinträgen abgefragt wird.
Es entspricht dabei einer SQL-Anweisung wie: Select Spalte1,SpalteN from Tabellenname where SpalteX=x and SpalteY=y

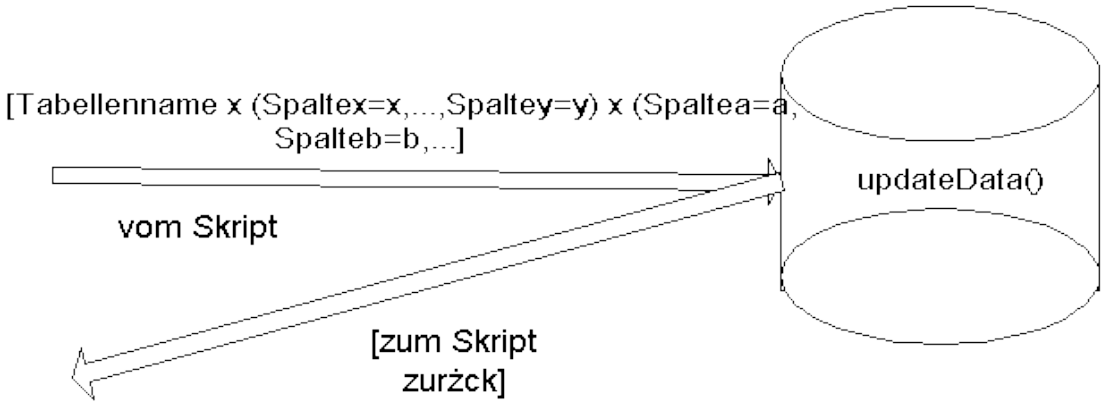


Abbildung 14 Update-Funktion von Datenbankeinträgen

UpdateData (Abbildung 14) aktualisiert Datenbankeinträge. Es entspricht etwa einem SQL-Statement wie:
Update Tabellenname set Spaltex=x and Spaltey=y where Spaltea=a and Spalteb=b,...

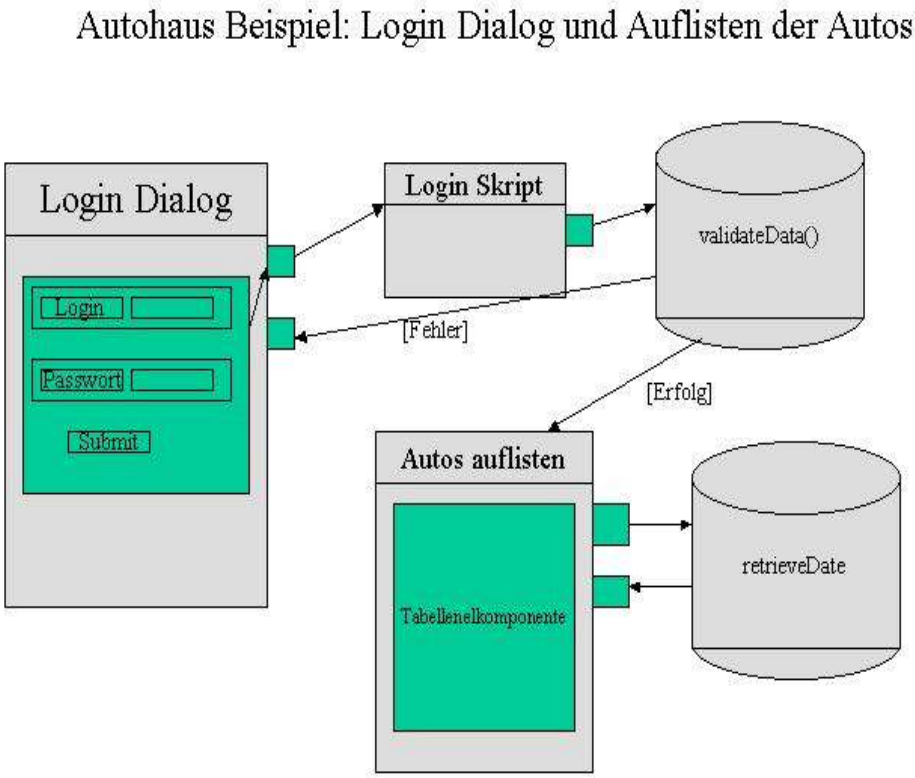


Abbildung 15 Beispiel für "PS-Autohaus"

Der Generator nimmt dem Benutzer viele Prüfungen von vornherein ab. Ist es also definiert, daß ein Name mind. 1 Zeichen haben muss und keine Zahlen enthalten darf, so generiert der Generator gleich eine entsprechende PHP-Überprüfung, die das übermittelte Feld auf korrekten Wert hin überprüft. Auch benutzerdefinierte Check-Regeln sind vorteilhafterweise nur einmal zu implementieren und dann auch verschiedene Eingabeelemente anwendbar.

4.2 Browser Navigations Knopf Problematik

In Webanwendungen benutzt der Anwender häufig auch die Browser typischen Navigations Elemente wie die Vorwärs- und Rückwärts Knöpfe, sowie den Aktualisierungs Knopf. Dies kann in der Webanwendung zu einem inkonsistenten Zustand führen. Außerdem kann es passieren, dass der Anwender den "Abschicken"-Knopf eines Formulars mehrfach klickt, falls nämlich die Antwortseite nicht schnell genug erscheint.

Um dieses Problem zu lösen, kann man, wannimmer ein Objekt gespeichert werden soll, eine eindeutige Objekt ID in die Session schreiben. Es wird dann zunächst geprüft, ob eine Objekt ID in der Session liegt. Falls ja - es wurde ein weiteres Mal auf den "Abschicken" Button geklickt oder der Browser führt eine "Aktualisierung" durch - wird das entsprechende Objekt in der Datenbank nur aktualisiert. Es wird also ein "update" Befehl durchgeführt.
Falls keine Objekt ID in der Session liegt, wird eine neue erzeugt und sowohl in der Session als auch in der Datenbank gespeichert ("insert" Befehl"). Die Objekt ID muß dabei eindeutig sein: es könnte beispielsweise ein Zeitstempel mit einer Benutzer ID kombiniert werden.
Falls der Benutzer ein neues Objekt speichern möchte, es liegt also schon eine Objekt ID in der Session, dann klickt er auf einen entsprechenden Link "Neues Objekt anlegen", dass auf ein Skript zeigt, dass zunächst alle Objekt IDs aus der Session löscht und anschließend eine neue Objekt ID anlegt und auf ein leeres Formular weiterleitet.

4.3 Die Lesezeichen Problematik

Es kann vorkommen, dass ein Benutzer einen Link zu einer bestimmten Seite der Webanwendung als Lesezeichen speichern möchte. Befindet sich diese Seite nun in einem geschützten Bereich, so würde bei erneutem Besuch dieser Seite eine Fehlermeldung angezeigt werden, falls der Browser in der zwischenzeit geschlossen wurde. Anstatt der Fehlermeldung kann natürlich die Login Seite angezeigt werden. Damit das System diese Seite nach dem Login direkt laden kann und somit nicht "vergisst", ist es nötig die Parameter der URL weiterzugeben. Speichert man also einen Link auf die Seite: http://www.-meinServer.de/shop/listAllItems.php?action=listAllItems,
so sollte das Skript listAllItems.php erkennen, dass der Benutzer nicht eingelogged ist und auf ein Login Skript weiterleiten. Dabei muß die gewünschte Aktion weitergegeben werden. Also zum Beispiel ein Redirect auf http://www.meinSer-ver.de/shop/login.php?action=listAllItems.
Das Login Skript führt die Benutzer/Passwort Verifikation durch und leitet dann wiederum an das Skript weiter, dass für die Aktion "listAllItems" zuständig ist.

Allgemein kann man sagen, dass man Parameter in der URL übergeben sollte, falls man Lesezeichen auf interne Seiten zulassen möchte.

4.4 Generierte Webanwendung

Als Ergebnis liefert der Generator eine Webanwendung, die vom Webanwendungsentwickler der selbigen nur noch auf einem Server eingebunden werden muss.
Der Generator übernimmt hier bereits die Aufgabe, die erzeugten PHP-Dateien sowie HTML- und sonstige Dateien zu trennen. So werden PHP-Include-Dateien in ein gesondertes Verzeichnis abgelegt, welches von außen nicht erreichbar ist. Mediaelemente sowie Stylesheets werden wiederum ebenfalls getrennt abgelegt, wie dies auch bei Bildern der Fall ist.
MySQL- und Installationsskripte werden dem Benutzer separat zur Verfügung gestellt, sodaß er diese nicht versehentlich ungeschützt veröffentlicht.

5 Technische Anforderungen

Technische Anforderungen aus Sicht des Entwicklers, der eine Webanwendung aus der im Vorfeld erstellten visuellen Umgebung generieren möchte:

- TCL/TK

Technische Anforderungen aus Sicht des Webseitenanbieters (der den Besuchern die Seite anbieten möchte):

- Webserver (z.B. Apache)
- PHP in der Version 4
- bei Nutzung von persistenten Daten:
    - PHP-Sessions und/oder MySQL-Server

Technische Anforderungen aus Sicht des Webanwendungsnutzer:

- HTML-Browser optional mit CSS-Unterstützung
- ggf. wird bei Nutzung von Cookies die Erlaubnis zur Nutzung von Cookies benötigt.

5.1 Unterstützung des Webseitenanbieters bei der Installation

Ziel der Anwendung soll ebenfalls sein, dem Webseitenanbieter beim Deployment, also der Installation seiner Webanwendung größtmöglich zu unterstützen.

Der Generator dieses Projekts soll hierfür folgende Unterstützung bieten:

- Erstellung von SQL-Skripten zur Einrichtung einer Datenbank
- Erstellung der passenden Verzeichnisstruktur mit logischen Verzeichnissen (Images, Stylesheets, PHP-Quellcode, HTML-Seiten)
- Automatische Erstellung von include-Dateien bei mehrfach benötigtem Code

6 Glossar

ELI:

Mit dem Eli-System kann man aus anwendungsnahen Spezifikationen vollständige Implementierungen von Sprachen und Übersetzungen in beliebige Zielsprachen automatisch generieren.

DEViL:

Ein Generator, der Entwicklungsumgebungen für visuelle Sprachen aus Spezifikatione generiert. Die generierte Umgebung enthält einen Struktureditor sowie einen Codegenerator.

HTML:

HTML (**H**yper**T**ext **M**arkup **L**anguage) ist eine Auszeichnungssprache (Markup Language). Sie hat die Aufgabe, die logischen Bestandteile eines textorientierten Dokuments zu beschreiben. Sie bietet die Möglichkeit an, Elemente eines textorientierten Dokuments, wie Überschriften, Textabsätze, Listen, Tabellen oder Grafikreferenzen, als solche auszuzeichnen.

Komponente:

Eine Komponente ist ein abstraktes Objekt mit einer grafischen Repräsentation und besitzt die Fähigkeit mit dem Benutzer zu interagieren. Beispiele für konkrete Komponenten sind: Schalter (Button), Eingabefeld, Tabelle.

PHP:

PHP ist eine serverseitig ausgeführte Skriptsprache für die Generierung von HTML-Dokumenten mit dynamischen Inhalt. PHP-Programme bzw. Skripte bestehen aus HTML-Dokumenten mit eingebetteten PHP-Anweisungen. Der umliegende HTML-Code ist statischer Inhalt des Dokumentes und wird unverändert übernommen bzw. an den Web-Server weitergereicht. Die PHP-Anweisungen werden durch den PHP-Interpreter ausgeführt und die dabei erzeugte Ausgaben werden an Stelle der Anweisungen in das Dokument eingefügt.

Programmstruktur:
Instanz des Sprachmodells. Die Programmstruktur ist also eine Datenstruktur, die den abstrakten Informationsgehalt eines bestimmten visuellen Programms repräsentiert. Äquivalent in textuellen Sprachen: Abstrakter Strukturbaum.

Rahmen:
Mit Hilfe von Rahmen kann der Anzeigebereich einer Webseite in verschiedene, frei definierbare Segmente aufgeteilt werden. Dabei verfügt jedes Segment über eigenen und von den anderen Rahmen unabhängigen Inhalt.

Spezifikation (DEViL):
Beschreibt das Sprachmodell, die visuelle Darstellung und Übersetzung in eine Zielrepräsentation.

Sprachmodell:
Modell der visuellen Sprache, das die Struktur der Sprachkonstrukte beschreibt, ohne auf deren visuelle Darstellung einzugehen. Wird in DEViL in Form einer ".model" Spezifikation angegeben. Äquivalent in textuellen Sprachen: Abstrakte Syntax.

Sprachkonstrukt:
Ein elementares Ausdrucksmittel der visuellen Sprache. Struktur und Kombinierbarkeit der Sprachkonstrukte ergibt sich aus dem Sprachmodell. Die grafische Repräsentation der Sprachkonstrukte kann z.B. durch visuelle Muster beschrieben werden.

SQL:
SQL (**S**tandard **Q**uerry **L**anguage) ist eine standardisierte Sprache für den Zugriff und die Manipulation von Datenbanken.

Visuelle Sprache:
Die zu entwickelnde visuelle Spezifikationssprache, mit der Webanwendungen spezifiziert werden können.

Visuelles Programm:
Instanz der visuellen Sprache. Das visuelle Programm ist also eine Spezifikation, aus der eine konkrete Webanwendung generiert werden kann.

Webseite:
Eine HTML-Seite, die im Browser dargestellt wird.

Webanwendung:
Sammlung von Webseiten, die mineinander, mit dem Benutzer, sowie mit verschiedenen Ressourcen auf einem Webserver, darunter auch mit Datenbanken, interagieren.

Webanwendungsentwickler:
Benutzt die zu entwickelnde visuelle Sprache, um Webanwendungen zu erstellen.

Webanwendungsnutzer:
Benutzt die vom Webanwendungsentwickler erstellte Webanwendung.