Fakultät für Elektrotechnik, Informatik und Mathematik
der Universität Paderborn

Projektgruppe
Generierung von Web-Anwendungen aus visuellen Spezifikationen

PaderWAVE

Zwischenbericht

Betreuer:
Dipl. Inform. Carsten Schmidt
Dr. Michael Thies

Teilnehmer:
Bastian Cramer
Oxana Paul
Jana Schäfer
Christian Schneider
Elvira Schumacher
Jens Siebert
Stephan Winter
Barbara Zimmermann

Paderborn, den 28. Januar 2005

Inhaltsverzeichnis

1 Einleitung

Der Zwischenbericht ist Bestandteil der Projektgruppe „Generierung von Webanwendungen aus visuellen Spezifikationen“ und beschreibt den aktuellen Entwicklungsstand der ersten Phase. Näheres zu dem zweistufigen Entwicklungsplan ist dem Pflichtenheft zu entnehmen. Der Zwischenbericht stützt sich auf den ersten Prototypen der zu entwickelnden visuellen Entwicklungsumgebung für Webanwendungen. Der Name der visuellen Entwicklungsumgebung lautet *PaderWAVE*.

Der Zwischenbericht ist wie folgt aufgebaut: Der zweite Abschnitt behandelt den Entwurf der visuellen Umgebung sowie den Entwurf der Sprache. Dabei werden die Ziele des zu entwickelnden Systems noch einmal kurz aufgeführt. Im dritten Abschnitt wird das Sprachmodell erläutert. Es werden die wichtigsten Sprachelemente erläutert, wobei der Nutzen für den Benutzer im Vordergrund steht. Auf ausführliche Erklärungen der einzelnen Attribute aller Sprachelemente wird bewusst verzichtet. Im vierten Abschnitt wird ein konkretes Beispiel vorgestellt, welches mit dem ersten Prototyp des zu entwickelnden Systems erstellt worden ist. Anschließend wird im fünften Abschnitt ein Ausblick auf das fertige System gegeben.

2 Die Entwicklung der visuellen Umgebung

Die Entwicklung der visuellen Entwicklungsumgebung lässt sich in zwei Hauptbestandteile aufsplitten. Zum einen muss die visuelle Umgebung selbst entwickelt werden, zum anderen muss das Sprachmodell entworfen werden. Bevor jedoch auf den Entwurf dieser beiden Teile eingegangen wird, soll zunächst der Nutzen und das Ziel des Systems im nachfolgenden Unterabschnitt verdeutlicht werden.

2.1 Ziele des Systems

Ziel der Projektgruppe ist die Entwicklung einer visuellen Entwicklungsumgebung, mit der Web-Designer Anwendungen für das Web visuell modellieren können, um danach den erforderlichen Anwendungs-Code ohne eigenes zutun von der Umgebung generieren zu lassen. Der Anwendungs-Code der generierten Webanwendung basiert auf HTML mit Anteilen an PHP-Skripten.

Durch die visuelle Modellierung und die Codeerzeugung durch die Umgebung ist kein teures Expertenwissen aus dem Bereich der Web-Programmierung erforderlich. Damit richtet sich die Umgebung vornehmlich an Web-Designer ohne Programmierkenntnisse.

Der Entwickler kann sich ganz auf den Inhalt der Webanwendung konzentrieren, ohne sich mit technischen Fragen der Umsetzung beschäftigen zu müssen.

Ein weiterer Vorteil für den Benutzer der visuellen Umgebung ist die Vermeidung von Programmierfehlern, die bei handgeschriebenen Webanwendungen schnell entstehen, insbesondere wenn diese komplexer werden. Die visuelle Umgebung erzeugt robusten und konsistenten Anwendungs-Code. Das Einschleichen von Fehlern in bestehenden Anwendungs-Code durch ständiges Modifizieren wird verhindert. Der Benutzer arbeitet nur mit dem Modell der Webanwendung. Aus diesem kann er einfach bestehende Sprachelemente entfernen oder das Modell um zusätzliche Elemente erweitern. Aus dem modifizierten Modell wird dann der Anwendungs-Code, nach erfolgreichem Bestehen der Konsistenzprüfungen, vollständig neu erzeugt. Dies verhindert die Erzeugung von fehlerhaftem Anwendungs-Code oder die Notwendigkeit, schlecht strukturierten Code erweitern zu müssen.

Durch die Reduzierung der Entwicklung auf die reine Modellierung der Webanwendung lassen sich in kürzester Zeit einfache Prototypen (Rapid-Prototyping) erstellen oder komplexe Anwendungen mit einer hohen Lebenszeit generieren.

Des Weiteren soll in der endgültigen Version dem Benutzer die Möglichkeit gegeben werden, selbst geschriebene Skripte in die generierte Webanwendung einbinden zu können. Dies gibt dem Benutzer die Freiheit, seine Webanwendung um nicht vorgesehene Funktionalität zu erweitern oder bereits bestehenden Anwendungs-Code wiederzuverwenden.

2.2 Realisierung mit DEViL

Die Entwicklung einer handgeschriebenen visuellen Entwicklungsumgebung ist eine aufwendige und fehlerträchtige Aufgabe. Insbesondere bei Änderungen oder Erweiterungen des Sprachmodells können sich schnell Fehler einschleichen.

Aufgrund der Komplexität des Sprachmodells kann der zugehörige Programmcode für den Entwickler nicht mehr durchschaubar sein und ist damit schwer wart- bzw. erweiterbar. Aus diesen Gründen bietet sich die Verwendung eines Generators an. In der Arbeitsgruppe „Programmiersprachen und Übersetzer“ von Prof. Dr. Uwe Kastens wird der Generator DEViL entwickelt.

Mit DEViL kann man visuelle Entwicklungsumgebungen aus Spezifikationen der Sprachstruktur und der grafischen Darstellung der Sprache generieren lassen. Dabei wird die Verarbeitung (Transformation) der Spezifikation (ausgehend vom abstrakten Strukturbaum) in eine Zielsprache separat spezifiziert. Zu Beginn wird das Sprachmodell der

visuellen Sprache spezifiziert. Die Spezifikation der Sprache basiert auf dem Konzept von Klassenhierarchien. Klassen des Sprachmodells enthalten Attributdefinitionen und können 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, in diesem Fall HTML mit eingebetteten Skriptsprachen-Anteilen (PHP), transformiert.

3 Die visuelle Sprache

Im Folgenden sollen nun die einzelnen Sichten der visuellen Umgebung erläutert werden.

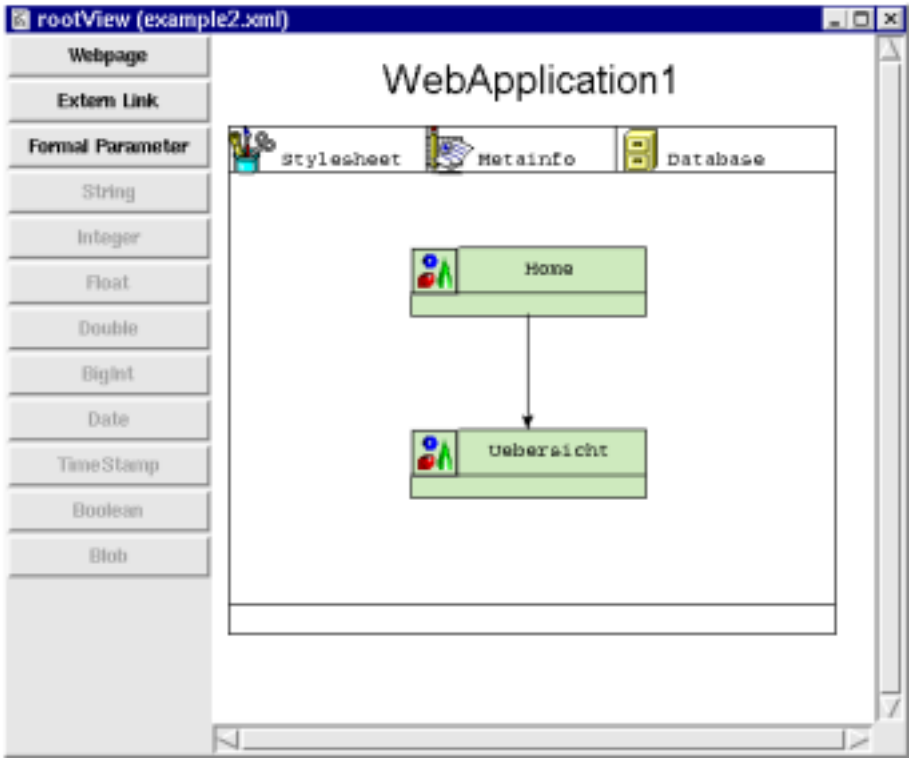

Abbildung 1: Die Webanwendungs-Sicht

3.1 Hauptsicht

In der Hauptsicht, der „rootView“, des PaderWAVE-Generators wird das Grundgerüst der Webanwendung modelliert. Beim Modellieren einer Webanwendung enthält die Hauptsicht im oberen Teil drei Bereiche (siehe Abbildung **??**): „Stylesheet“, „Metainformation“ und „Database“. Im unteren Bereich können auch *externe Links* eingefügt werden. Im linken Bereich der „rootView“ (grau markiert) befinden sich *Buttons* für die Darstellung von Webpages, externe Links, formalen Parametern und entsprechende Typen dazu.

In dem unteren abgegrenzten Bereich können die externen Links aller Webseiten aufgelistet werden. Diese Liste wird aus Gründen der besseren Übersicht dargestellt. Hier wird der Name des Links und die zugehörige Verlinkung mit Hilfe von *Properties* dargestellt, zum Beispiel **Google -> http://www.google.de**.

In der Mitte der Hauptsicht können mehrere Webseiten mit dem Button „Webpage“ hinzugefügt werden. Mittels Properties können einzelnen Webseiten ein Namen und Stylesheetblock zugeordnet werden. Des Weiteren können die Formalparameter mit dem Button „Formal Parameter“ in jede Webseite eingefügt werden. Jede Webseite besitzt eine eigene Sicht „WebPageView“. Dies bedeutet, dass jede Webseite einzeln bearbeitet werden kann.

In der Hauptsicht werden außerdem die Verbindungen zwischen den einzelnen Webseiten angezeigt (siehe Abbildung **??**). Sobald eine Webseite ein Formular enthält, wird ein Pfeil angezeigt, der von der Webseite ausgehend auf dieselbige (Schlinge) zeigt. Um eine Verbindung zu einer anderen Webseite darzustellen, muss in der „webpageView“ der ersten Webseite, über Properties des Formulars, die „action“ der gewünschten zweiten Webseite ausgewählt werden.

3.2 Sicht auf eine Webseite

Beim Klicken auf eine Webseite öffnet sich die „webpageView“ (siehe Abbildung **??**). Eine Webseite kann unterschiedliche Komponenten besitzen, welche links in der Sicht mit Hilfe von Buttons dargestellt werden. Einige Buttons werden ausgegraut angezeigt, weil diese erst nach Einfügen von bestimmtem Komponenten aktiviert werden. So werden zum Beispiel die Elemente „Row“ und „Column“ erst nach Einfügen von „Table“ aktiv. Dabei kann der Benutzer diese Komponente, „Row“ und „Column“, nur in der zulässigen Container-Komponente „Table“ einfügen.

Es folgt die Beschreibung der einzelnen Komponenten. Diese werden mittels Properties bearbeitet. Zur besseren Verteilung von Komponenten/Elementen auf einer Web-

Abbildung 2: Die Webpage-Sicht

seite kann eine Tabelle (Button „Table“) zur Hilfe genommen werden. Diese dient der Anordnung von Elementen und ist in der fertigen Webanwendung nicht sichtbar. In den eingefügten Zeilen und Spalten der Tabelle können pro Zelle mehrere Komponenten platziert werden, dies kann wiederum eine Tabelle sein. Mit dem Button „Heading“ können die Überschriften in verschiedenen Größen dargestellt werden. Des weiteren können Texte (Button „Text“), die Bilder (Button „Image“) und Formulare (Button „Form“) hinzugefügt werden. Ein Formular kann mehrere Komponenten, wie Eingabefeld (Button „Text entry“), Passwortfeld (Button „Password entry“), Textarea (Button „Textarea entry“), Checkboxen (Button „Checkbox“) und Radiobutton-Gruppe (Button „Radio group“) enthalten. Einer Radiobutton-Gruppe, die zu Beginn nur aus zwei Radiobuttons besteht, können weitere Radiobuttons mit Hilfe des Buttons „Radio button“ hinzugefügt werden. In einer Radiobutton-Gruppe kann nur ein Radiobutton mittels Properties selektiert werden. In eine Webseite kann auch ein *TableOutput* - Element eingefügt werden. In dieser Tabelle wird die Ausgabe einer Abfrage aus der Datenbank angezeigt. Hier werden die *DataSourceActualParameter* festgelegt. Außerhalb der Webseite (unten) werden auch die

angelegten Formalparameter aus der Hauptsicht dargestellt. Diese sind nur zur Übersicht da. Um die formalen Parameter bearbeiten zu können, muss man in die Hauptsicht wechseln.

3.2.1 Komponenten einer Webseite

Eine Webseite wird aus einzelnen Elementen zusammengesetzt. Diese Elemente, in PaderWAVE als Komponenten bezeichnet, entsprechen zunächst den Elementen, die auch in HTML zur Erstellung einer Webseite verwendet werden können. Dies sind zum Beispiel Bilder, Tabellen, Überschriften oder Textblöcke. Die entsprechenden Komponenten in PaderWAVE für diese Elemente werden 1:1 auf die HTML-Elemente abgebildet.

Zur erweiterten Gestaltung einer Webseite wurde die Frame-Komponente eingeführt. Mit Hilfe dieser Komponente soll es dem Benutzer möglich sein, Komponenten zu gruppieren und in einer bestimmten Reihenfolge oder Position auf der Webseite anzuordnen. Ein Frame kann beliebige andere Komponenten enthalten. Insbesondere ist es auch möglich, beliebig viele Frames zu verschachteln um komplexe Layouts zu ermöglichen.

Um die Eingabe von Daten zu ermöglichen, wurde eine Formular-Komponente eingeführt. Diese Komponente entspricht dem HTML-Formular und kann unterschiedliche Formular-Elemente enthalten. Es handelt sich bei diesen Elementen um ein- oder mehrzeilige Texteingabefelder, Passwort-Felder in denen der eingegebene Text durch Stern-Symbole ersetzt wird, Checkbox-Elemente sowie Submit- und Reset-Schaltflächen. Eine besondere Klasse dieser Formular-Elemente stellen die aus HTML bekannten Radio-Buttons dar. Das sich diese Elemente in Gruppen zusammenfügen lassen, um zum Beispiel Multiple-Choice-Antworten auf eine Frage zu ermöglichen, musste in PaderWAVE auch dieses Konzept berücksichtigt werden. Dies wurde mit Hilfe des Elementes RadioButtonGroup erreicht, in dem sich beliebig viele Radio-Buttons zu einer Gruppe zusammenfügen lassen. Zur Zeit sind einige der HTML-Fomular-Elemente noch nicht in PaderWAVE realisiert. Dies gilt zum Beispiel für Auswahllisten.

Um in einer Webseite Bilder verwenden zu können muss das Modell der Webanwendung zunächst gespeichert werden. Danach kann über Properties der Webanwendung der Pfad des Wurzelverzeichnisses der Bilder gesetzt werden. Erst dann kann in der „WebpageView“ ein Bild über Properties des Bildes ausgesucht werden.

Weiterhin war es nötig einige zusätzliche Komponenten zu entwerfen, die keine Entsprechung in HTML haben. Hierzu gehören die OutputElement-Komponenten. Diese dienen dazu, Daten auf eine Webseite darzustellen, die zum Beispiel aus einer Datenbank-Abfrage stammen.

3.2.2 Übergabe von Parametern zwischen Webseiten

Die Modellierung der Parameterübergabe aus einem Formular zur weiteren Verarbeitung auf einer Zielseite lehnt sich an die Parameterübergaben, wie sie in Programmiersprachen zu finden sind, an. Es existieren auch hier Konzepte, wie formale und aktuelle Parameter. Als einfaches Beispiel soll hier die Üergabe eines Wertes aus einem Textfeld auf einer Quellseite an eine Zielseite dargestellt werden. Der übergebene Wert wird auf der Zielseite ausgegeben.

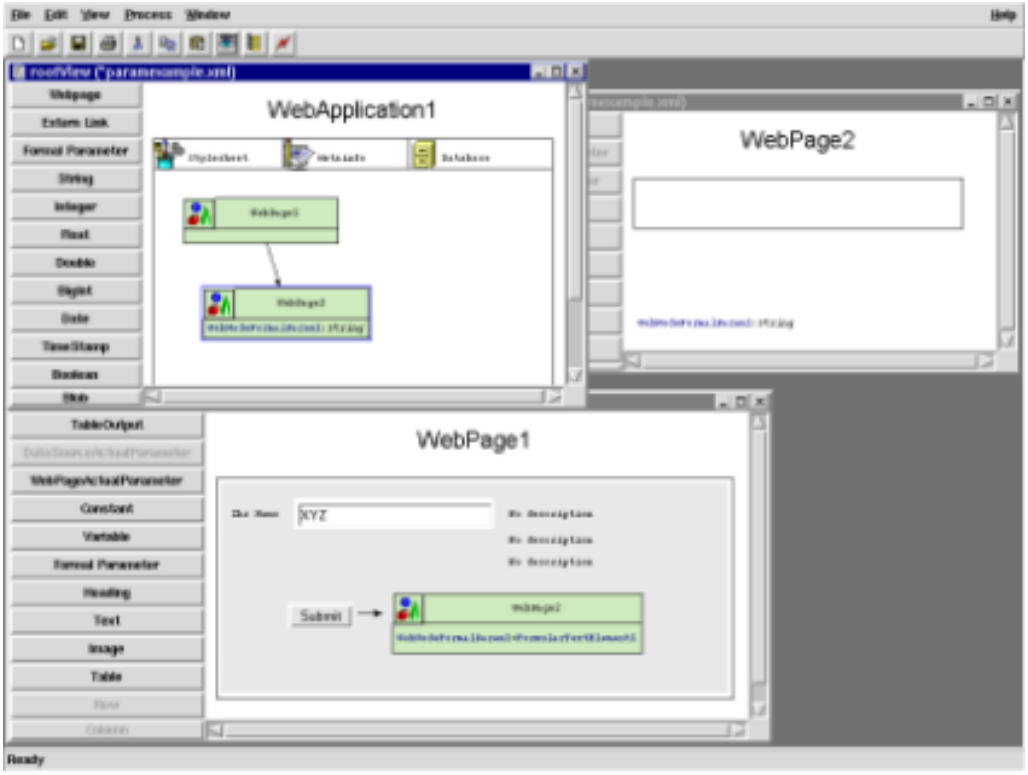

Abbildung 3: Modellierung der Parameterübergabe zwischen einer Quellseite (WebPage1) und einer Zielseite (WebPage2)

Zunächst muss der Anwender auf der Zielseite ein Element vom Typ OutputElement einfügen, welches die Ausgabe des übergebenen Wertes realisiert. Als Inhalt dieses Elementes wird ein formaler Parameter definiert. Dadurch wird der an diesen formalen Parameter übergebene Wert in dem OutputElement ausgegeben.

Um nun die Übergabe eines Wertes an den formalen Parameter zu modellieren, muss der Benutzer auf der Quellseite ein Formular-Element in ein Formular einfügen. Dies kann zum Beispiel ein Element vom Typ FormularTextElement sein, welches ein Textfeld

darstellt. Die eigentliche Wertübergabe wird nun dadurch modelliert, dass der Anwender im Formular einen aktuellen Parameter angibt. Dieser aktuelle Parameter verknüpft das Formular-Element und den formalen Parameter. Durch diese Verknüpfung wird bei der Codegenerierung der entsprechende Code zur Auswertung eines übergebenen Parameters in die Zielseite eingefügt.

Der Wert eines Parameters muss allerdings nicht zwangsläufig aus einem Formular-Element stammen. Es ist auch möglich, einen konstanten Wert zu übergeben. Weiterhin soll es in späteren Versionen von PaderWAVE möglich sein, dass beliebig viele Quellseiten auf eine Zielseite verweisen können. Derzeit ist dort nur eine 1:1-Beziehung möglich.

3.3 Datenbank und Datenquellen

Beim Klicken auf „Database“ in der Hauptsicht wird die Sicht „DatabaseView“ geöffnet, in der die Tabellen einer Datenbank und zugehörige Filter angelegt werden können. Des Weiteren können dort die Zugriffsparameter für die Datenbank festgelegt werden. Auf die Definition und Verwendung von Datenquellen soll nun im Folgenden näher eingegangen werden.

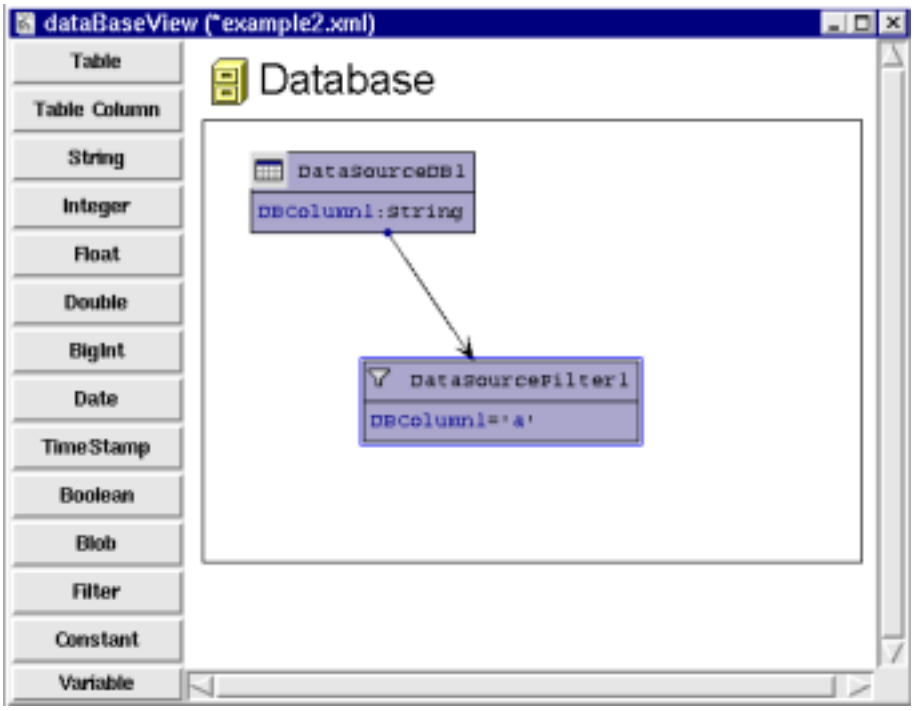

Abbildung 4: Die Database-Sicht

Eines der primären Ziele von Web-Anwendungen ist es, dynamischen Inhalt aus di-

versen Datenquellen darzustellen. Dazu ist es zunächst nötig, anzugeben, woher die Daten stammen sollen. Im ersten Prototyp von PaderWAVE gibt es hier die Möglichkeit, relationale Datenbanken einzubinden. Später soll es auch möglich sein, zum Beispiel Textdateien als Datenquellen anzusehen und einzubinden.

Um nun konkret Datenquellen zu definieren, muss erst in die bereits oben erwähnte *DataBase-View* gewechselt werden. Mit einem rechten Mausklick können die globalen Einstellungen für die Datenbank angegeben werden. Diese Angaben werden später für den PHP Interpreter benötigt, um auf die Datenbank zuzugreifen.

In der gerade definierten Datenbank können nun mit Hilfe des “Tables-Buttons“ einzelne Datenquellen definiert werden. Datenquellen in PaderWAVE repräsentieren dabei eine Liste von Tupeln gleicher Struktur, beispielsweise in folgender Form:

```
{ (1,"BMW","150 PS,6 Zylinder"),
  (2,"Mercedes","180 PS,6 Zylinder"), ... }
```

Mit Hilfe von Datenquellenfiltern können dann Listenelemente selektiert werden oder mehrere Datenquellen zusammengeführt werden. Außerdem können einzelne Tupelelemente ausgeblendet werden, ähnlich der Projektion in relationalen Datenbanken. Im ersten Prototyp von PaderWAVE beschränken wir uns jedoch auf das Filtern einer einzigen Datenquelle.

Diese Datenquellen können prinzipiell auch als Tabellen, wie sie in einer relationalen Datenbank existieren, angesehen werden. Sie bestehen aus beliebig vielen Spalten, denen jeweils ein in der Tabelle eindeutiger Name und ein Typ zugeordnet werden.

Um eine Spalte hinzuzufügen wird der “Table Column Button“ geklickt und anschließend können der Spalte ein Name und ein Typ zugeordnet werden. Als Typen stehen die gängigsten Typen der relationalen Datenbank *MySQL* zur Verfügung: *String, Integer, Float, Double....*

Wenn die Datenquellen vollständig definiert sind, wird ein PHP Skript “dbinit.php“ erzeugt, dass, wenn es auf einen Webserver geladen und ausgewählt wird, das visuell erstellte Datenbankschema automatisch erstellt. Nun fehlen noch Filter um auf bestimmte Teilmengen der Datenquellen zuzugreifen. In *MySQL* würde man diese Filter als SQL Abfragen ansehen. Zum Beispiel in der Form

```
select * from Kunden where Kundenname='Meier'
```

um die Kundendaten des Kunden mit Namen “Meier“ zu bekommen. In PaderWAVE wird so eine Anfrage als Filter angesehen, der ebenfalls in der *DataBase View* mit dem

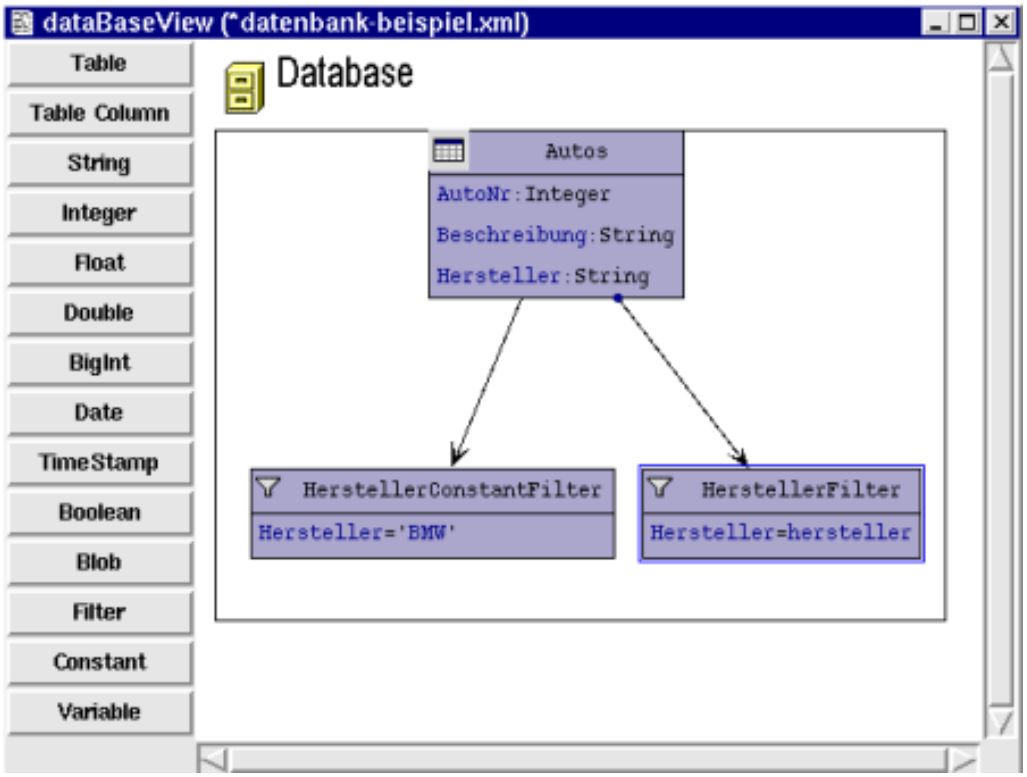

Abbildung 5: Eine Datenbank Tabelle “Autos“ mit den Spalten *AutoNr, Beschreibung* und *Hersteller*. Darunter zwei Filter. Der linke filtert die Tabelle “Autos“ konstant nach “BMW“, der rechte filtert die *Hersteller* Spalte variabel.

“Filter-Button“ angelegt werden kann. Dieser Filter muss zunächst in den Properties mit einer bereits existierenden Datenquelle verbunden werden, die gefiltert werden soll. In der visuellen Darstellung erscheint nun ein Pfeil von der Datenquelle zum Filter. Dies soll den Datenfluss verdeutlichen. Als nächstes müssen ein Spaltenname und ein Operator ausgewählt werden. Der Spaltenname entspricht dabei dem Attribut, auf das der Filter angewendet werden soll. Im obigen Beispiel wäre das also “Kundenname“. Der Operator ist der Vergleichsoperator, im obigen Beispiel also das Gleichheitszeichen. Als letztes muss der Ausdruck angegeben werden, mit dem gefiltert werden soll. Dabei gibt es zwei Möglichkeiten: Es kann entweder ein konstanter Ausdruck angeben werden, symbolisiert durch den “Constant Button“ oder es kann ein variabler Ausdruck angegeben werden, der durch den “Variable-Button“ erstellt werden kann. Ein konstanter Ausdruck würde immer nach einem vordefinierten Wert filtern, egal, welche Eingabe die Filter Funktion bekommt. Zum Beispiel würde

```
select * from Kunden where Kundennummer=1
```

immer nach der Kundennummer 1 filtern. Ein variabler Ausdruck kann hingegen einen beliebigen Wert als Eingabe bekommen und danach filtern. Der Filter wäre somit eine Funktion der Form:

```
function filter ($VariableExpression){
  select * from DataSource where
         DataSource.Column [Operator] $VariableExpression
}
```

Intern wird ein variabler Ausdruck, also ein Aktualparameter, an den formalen Parameter des Filters gebunden. Als aktueller Parameter kann dann zum Beispiel ein formaler Parameter einer Webseite eingesetzt werden. Hier kann die oben beschriebene Parameterübergabe als Eingabe für den Filter benutzt werden.

Den Filter als Funktion anzusehen hat außerdem den Vorteil, dass in der Methodensignatur beliebig viele formale Parameter übergeben werden können, die im Rumpf unterschieden werden können. PaderWAVE bietet mit diesem Konzept prinzipiell schon die Möglichkeit Filter zu verknüpfen, was jedoch im ersten Prototyp noch nicht implementiert ist.

Im ersten Prototyp wird bisher nur eine Ausgabe von Datenquellen ermöglicht. Dazu gibt es ein Sprachelement “TableOutputElement“. Dies benötigt eine Referenz auf einen Filter und übergibt diesem dann einen Ausdruck, der wie oben erwähnt entweder konstant sein kann oder sich als variabler Ausdruck zum Beispiel auf einen übergebenen Parameter beziehen kann.

Des Weiteren ist es bisher nicht möglich Daten in den Datenquellen zu speichern oder mehrere Datenquellen zusammenzusetzen, dazu jedoch mehr in Kapitel **??**.

3.4 Stylesheets

Beim Klicken auf den Bereich „Stylesheet“ öffnet sich die Sicht „stylesheetView“, in welche der Benutzer Formatvorlagen für komplette Webseiten und/oder einzelne Elementen von Webseiten definieren kann. Hier können zunächst *Stylesheetblöcke* mit dem Button „Block“ erzeugt werden, welche dann von den Webseiten bzw. Elementen referenziert werden können. In den Stylesheetblöcken kann der Benutzer die Darstellung in Form von Attribut-Werte-Paaren mit dem Button „Attribute/Value Pair“, zum Beispiel **backgroundcolor = white**, spezifizieren.

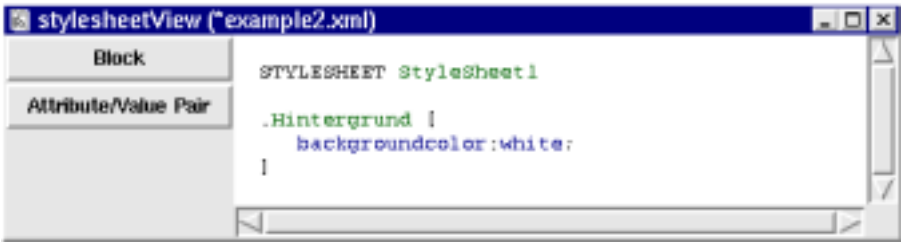

Abbildung 6: Die Stylesheet-Sicht

3.5 Meta-Informationen

Unter „Meta-Information“ in der Sicht „metaInformationView“ kann der Benutzer Meta-Informationen definieren, mit denen die Webanwendung mit zusätzlichen Informationen angereichert werden kann. Die Meta-Informationen können zum Beispiel von Suchmaschinen verwendet werden, um ein besseres Profil über den Inhalt der Webseiten erstellen zu können. Mit dem Button „Attribute/Value Pair“ können die Meta-Informationen in Form von Attribut-Werte-Paaren, zum Beispiel **Autor = Tom+Jerry** angelegt werden.

Abbildung 7: Die Metainformationensicht

4 Konkretes Beispiel

Im folgenden Beispiel wurden nahezu alle verfügbaren und derzeit generierbaren Fähigkeiten ausgenutzt. Dennoch wurde das Beispiel bewusst kleinstmöglich gehalten, um es später (auch trotz Änderungen) noch weiternutzen zu können. In Abbildung **??** ist die Hauptsicht der Beispielanwendung abgebildet.

Das Beispiel besteht aus folgenden Seiten:

In Abbildung **??** ist eine Übersichtsseite zu sehen. Sie bietet eine Tabelle, Bilder zweier Autos, eine Überschrift, einen externen Link zu Google sowie Text, dem bereits Styles-

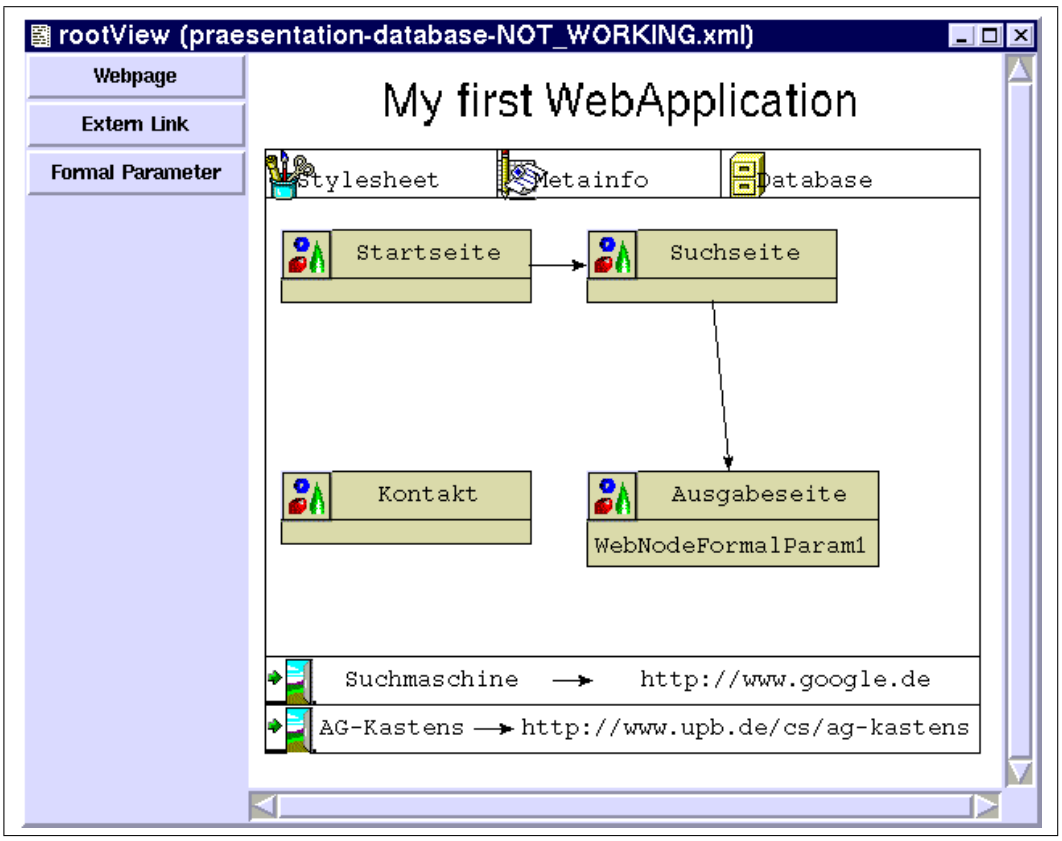

Abbildung 8: Die RootView der Beispielanwendung

heets zugeordnet wurden. Von der Startseite aus kann man über den Link Fahrzeugsuche zu eben dieser navigieren.

Auf der Suchseite (vgl. Abb. **??** befindet sich ein einzelnes Formular sowie ein Link auf die Kontaktseite. Das Formular ist an eine Datenbank angebunden, aus der Autos sowie deren Beschreibungen bezogen werden. Über ein (ebenfalls generiertes) Skript (dbinit.php) wurde die Datenbank bereits initialisiert, Beispieldaten wurden per Hand eingetragen. Durch die Eingabe eines vorhandenen Auto-Typs wird nun durch eine Datenbankabfrage mit vorheriger POST-Variablenübergabe die Zielseite generiert. Die Zielseite zeigt nun die in der Datenbank enthaltenen Autos vom angegebenen Typ an (vgl. Abb. **??**).

Herzlich Willkommen!

Herzlich Willkommen im Autohaus Futurecars.

Hier finden Sie absolut schnellsten und neuesten Autos.

Hier der neue A4: Und hier die Zukunft:

Legen Sie nun gleich mit unserer Fahrzeugsuche los:

Fahrzeugsuche

Es gefällt Ihnen hier nicht? Dann suchen Sie doch bei:

Google

Abbildung 9: Die Startseite der Beispielanwendung

Suche

Submit

Die große Auto-Suche

Sie haben ihr Auto nicht gefunden? Dann sprechen Sie uns direkt an.

Abbildung 10: Die “Suche-Seite“ der Beispielanwendung

3 Audi A6 3.0 TDI
6 Audi TT 1.8T

Abbildung 11: Die “Ergebnis-Seite“ der Beispielanwendung

5 Ausblick

Mit der Fertigstellung des ersten Prototyps konnte bereits eine Vielzahl der geplanten Features umgesetzt werden. Die wichtigsten Sprachelemente zur Modellierung einer kompletten Webanwendung sind vorhanden. Bei der Entwicklung des ersten Prototyps konnten wir wertvolle Erfahrungen und weitere Ideen sammeln, die wir in die Entwicklung der endgültigen Version einfließen lassen. Neben der Verfeinerung vorhandener Sprachelemente werden dem Sprachmodell weitere Sprachelemente hinzugefügt, um die im Vorfeld geplanten Konzepte umsetzen zu können. Das WebZone-Konzept fehlt in dem ersten Prototyp noch vollständig. Das Sprachmodell enthält zwar schon das Sprachelement WebZone, allerdings ohne die gewünschte Funktionalität. Das Sprachelement ist zentraler Bestandteil der Modell-Hierarchie einer Webanwendung und wurde deshalb im ersten Prototyp bereist eingeführt, um die spätere Entwicklung der fertigen Version zu erleichtern. Die Umsetzung von WebZone's ist somit eine wichtige Teilaufgabe bei der Weiterentwicklung des Prototyps. Die WebZone's dienen dazu, geschützte Bereiche in einer Webanwendung zu definieren. Eine WebZone besteht zum einen aus einer Menge von Webseiten und einer Kontroll-Liste. In der fertigen Webanwendung darf ein Benutzer nur dann eine Webseite aufrufen, wenn er die Zugriffsrechte der Webseite zugehörigen WebZone besitzt. Beim ersten Aufruf einer geschützten Webseite werden die Zugriffrechte des Benutzers mit Hilfe der Kontroll-Liste überprüft. Hat er die entsprechenden Rechte, wird die ursprünglich vom Benutzer angeforderte Webseite dargestellt. Falls nicht, wird auf eine entsprechende Fehlerseite verwiesen.

Ein weiteres fehlendes Feature ist die Änderung / das Einfügen von Datenbank-Einträgen. In der aktuellen Version kann nur lesend auf Einträge von Datenbank-Tabellen zugegriffen werden. In der endgültigen Version soll der Web-Entwickler die Möglichkeit haben, Eingabe-Formulare zu definieren, mit denen der Web-Benutzer Daten übermitteln kann und diese persistent in einer Datenbank gehalten werden.

Des Weiteren soll dem Benutzer die Möglichkeit gegeben werden, eigene PHP-Skripte in die Webanwendung einzubinden. Damit kann er seine Webanwendung um spezielle Funktionalität in Form von Skripten erweitern. Dies vergrößert die Einsatzmöglichkeiten der visuellen Entwicklungsumgebung und bietet zudem dem Benutzer die Weiterverwendung vorhandener Skripte. Um die bestehende Funktionalität der vorhandenen Sprachelemente mit benutzerspezifischen Anwendungs-Code zu verknüpfen, müssen geeignete Schnittstellen geschaffen werden, um eine korrekte Parameterübergabe zu gewährleisten. Eine dieser Schnittstellen wird das neue Sprachelement *WebAction* sein, mit dem der

Benutzer die Möglichkeit haben wird, Anwendungs-Code in seine Webanwendung zu integrieren. Das WebAction-Element referenziert dazu wiederum ein Skript, welches zwei mögliche Ergebnisse (true/false) zurückliefert. Je nach Ergebnis wird auf eine der vom Web-Entwickler angegebenen Webseiten gesprungen.

Die Erzeugung von korrektem und robustem Code ist ein entscheidender Punkt für die Verwendbarkeit eines Generators. Wird fehlerhafter Code erzeugt, hat der Web-Entwickler nur die Möglichkeit den generierten Code manuell zu korrigieren. Dies ist mühselig und macht den Einsatz des Generators fragwürdig. Deshalb ist es wichtig, möglichst früh Fehler zu erkennen. zunächst müssen Modellierungsfehler seitens des Benutzers ausgeschlossen werden. Das Erstellen eines invaliden Modells muss von Beginn an verhindert werden. Die fertige Version der visuellen Umgebung muss daher um entsprechende Validierungs-Funktionen erweitert werden, mit denen zum Beispiel das Einhalten von Wertebereichen geprüft werden kann. Bei der Modellierung lassen sich nicht alle Typen von Fehlern erfassen, zum Beispiel das Fehlen von Referenzen. Deswegen müssen bei der Erzeugung des Anwendungs-Codes weitere Überprüfungen vorgenommen werden, wie zum Beispiel das Verwenden von nicht vollständig definierten Sprachelementen. Eine weitere Fehlerquelle sind Eingabefehler zur Laufzeit der Webanwendung. Dies können zum Beispiel falsche Eingaben in Formularen sein. Der Benutzer gibt Text ein, es wird aber ein Wert erwartet. Diese Fehler müssen durch einen möglichst robusten Anwendungs-Code, welcher von dem Generator erzeugt wird, abgefangen werden.

Bei der Modellierung erster Webanwendungen mit Hilfe des ersten Prototyps haben sich einige Konzepte als verbesserungswürdig herausgestellt. Das Erstellen von StyleSheets verlangt vom Web-Entwickler noch zu viel Html-Kenntnisse. Durch die Eingabe von Attribut/Werte-Paaren muss er die genauen Attribut-Bezeichnungen auswendig kennen. Das Konzept soll verbessert werden, indem eine Auswahl an gültigen Attribut-Bezeichnungen angezeigt wird. Der Benutzer muss nur noch das gewünschte Attribut auswählen. Nach diesem Prinzip soll auch das Definieren von Meta-Informationen erleichtert werden.

Zusätzlich soll dem Benutzer die Arbeit erleichtert werden und die Entwicklung von Webanwendungen beschleunigt werden, indem viel Funktionalität in Form einer ToolBox bereitgestellt werden soll. Diese soll unter anderem fertige Skripte enthalten, die der Entwickler in seiner Webanwendung verwenden kann. Darüber hinaus soll bei Bedarf eine Navigationsleiste, welche Verweise auf ausgezeichnete Webseiten enthält, automatisch generiert werden und in den Webseiten eingebunden werden.