Webtechnologien All in One

Eine praxisorientierte Einführung in moderne Webtechnologien

Jörg Barres

9. Dezember 2014

Inhalt

Vorwort

Liebe Leserin, lieber Leser,

der Gedanke zu diesem Buch kam mir, als ich für eines meiner IT-Trainings keine passende Lektüre gefunden habe.

Die einzigen Bücher, die alle notwendigen Themen abdeckten, waren zum einen ohne Neuauflage seit vielen Jahren auf dem Markt und deshalb vergriffen und zum anderen so umfangreich, dass sie sich zwar gut als Nachschlagewerk, nicht jedoch als begleitende Lektüre zu einer Veranstaltung eigneten.

Also beschloss ich, ein Buch zu schreiben, mit dem Sie sich in recht kurzer Zeit ein umfangreiches Wissen über moderne Webtechnologien selber aneignen können.

Dieses Buch erhebt nicht den Anspruch, Sie mit allen Technologien bis in die Tiefe vertraut zu machen. Mein Ziel ist es vielmehr, mit Ihnen anhand eines kleinen Web-Projektes die meisten der heute üblichen Techniken zu erarbeiten.

Wenn Sie dieses Buch durchgearbeitet haben, werden Sie in der Lage sein, selber zu entscheiden, welche der Techniken Sie durch weiterführende Literatur oder Veranstaltungen vertiefen möchten, welche Vor- und Nachteile die jeweiligen Technologien haben und mit welchem Aufwand Sie bei der Entwicklung eigener Projekte rechnen müssen.

Mir ist natürlich klar, dass die Entwicklung von Webseiten durch den Einsatz passender Entwicklungsumgebungen massiv erleichtert werden kann. Dennoch verzichte ich in diesem Buch darauf, denn so praktisch diese Entwicklungsumgebungen auch sein mögen, am Anfang müssen Sie sich erst einmal viel Wissen aneignen, um mit ihnen umgehen zu können. Und das lenkt von den eigentlichen Themen dieses Buches ab. Deshalb empfehle ich Ihnen, nur einen einfachen Editor zu verwenden. Sehen Sie sich dazu auch Abschnitt B im Anhang an.

Vielleicht bin ich da auch etwas altmodisch, aber Code, den Sie selber in einem Editor geschrieben haben, in dem Sie selber die Fehler gesucht und gefunden haben, in dem Sie jedes Paar Klammern und jede Einrückung selber vorgenommen haben, diesen Code haben Sie verstanden. Und dann können Sie immer noch auf eine Entwicklungsumgebung umsteigen, sich über die vielen Arbeitserleichterungen freuen und über die leider auch immer vorhandenen Einschränkungen ärgern.

Dieses Buch gliedert sich in drei wesentliche Teile. Im ersten Teil, dem *Fundament*, werden Sie die die Technologien HTML, PHP und CSS kennenlernen und eine erste strukturelle Umsetzung des Projektes erstellen. Im zweiten Teil, dem *Rohbau*, werden Sie die Funktionalität der einzelnen Seiten erstellen und dazu Formulare sowie die Datenbanksprache SQL kennenlernen. Und im dritten Teil, der *Fertigstellung*, werden

Sie Javascript und das Javascript-Framework jQuery kennlernen, um den Benutzerkomfort der Seiten zu steigern zusätzliche Funktionalitäten zu ermöglichen.

Ein paar Worte zum Stil des Codes

Jeder Programmierer hat seinen eigenen Stil. Das ist auch gut so, denn sonst wäre die Programmierung von Anwendungen nur halb so kreativ. Allerdings gibt es durchaus guten und schlechten Code, wobei die Meinungen hier natürlich ebenfalls auseinandergehen.

Meiner ganz privaten Meinung nach kennzeichnet vor allem die Lesbarkeit einen guten Programmierstil. Es ist heute nicht mehr notwendig, extrem kompakten Code zu erstellen, wie das noch in den 80er Jahren oft nötig war. Ebenso ist es (meistens) nicht mehr zwingend notwendig, optimal performanten Code zu erstellen. Die moderne IT verfügt über so hohe Ressourcen, dass wir uns durchaus etwas Platz- und Rechenzeitverschwendung leisten können. Von daher habe ich in diesem Buch Code dargestellt, der sicher Optimierungspotential aufweist. Aber dafür ist er einfach zu lesen und zu verstehen.

Programmcode im Text?

Ich habe lange überlegt, ob es Sinn macht, den Programmcode ebenfalls in den Text zu integrieren. Schließlich liegt Ihnen ja der Quellcode in seinen verschiedenen Inkarnierungen vor (Download siehe Anhang B.1). Ich habe mich aber dennoch entschlossen, wesentliche Teile des Codes in den Text aufzunehmen, denn vielleicht wollen Sie das Buch ja nicht gerade am Schreibtisch lesen oder haben keinen Laptop neben Ihrem Sessel stehen. Außerdem erkläre ich viele Zeilen des Codes und nehme dabei Bezug auf die abgedruckten Zeilennummern, so kann ich sicher sein, dass Sie den richtigen Bezug haben. Und schließlich ist es etwas anderes, gedruckten Code zu lesen und zu verstehen, als Code, der Ihnen am Bildschirm in einem Editor angezeigt wird.

Und nun wünsche ich Ihnen viel Vergnügen beim Einstieg in die Webtechnologien.

Jörg Barres, im Sommer 2014

Teil I.

Das Fundament

1. Das Projekt FahrFlott

1.1. Das Pflichtenheft

Der Fahrzeugverleih FahrFlott GmbH benötigt zur besseren Verwaltung seines Fuhrparks sowie einer leichteren Kunden- und Rechnungsverwaltung eine Web-basierte Lösung. Diese soll einen öffentlich zugänglichen Bereich für Online-Besucher sowie einen abgesicherten Verwaltungsbereich haben.

1.1.1. Die Gestaltungsvorgaben

Die FahrFlott GmbH hat sich für die Gestaltung der Website von einem Grafiker einen Entwurf fertigen lassen. Dieser ist in Abbildung 1.1 zu sehen.

Das groß dargestellte Bild eines Sportwagens soll beim Öffnen der Seite animiert erscheinen, ebenso der Logo-Schriftzug im Kopfbereich.

Selbstverständlich sind moderne Designaspekte wie abgerundete Ecken oder animierte Effekte, etwa beim Überfahren mancher Elemente mit der Maus, ebenso zu berücksichtigen.

Abb. 1.1.: Die Designvorgabe

Gestalterisch teilt sich die Website in vier Bereiche auf, den Kopfbereich mit dem Logo der FahrFlott GmbH, den Fußbereich mit Angaben zum Copyright, den AGBs und dem Impressum, einen linken Bereich für die Navigation und die Anmeldung sowie schließlich noch einen Inhaltsbereich.

Besonders wichtig ist der Einsatz der Firmenschriftart „Pacifico“, die sowohl im Logotext als auch für Überschriften eingesetzt werden soll.

1.1.2. Der öffentliche Bereich

Für den Besucher sollen auf der Website die wichtigsten Informationen über die FahrFlott GmbH zu finden sein. Das beinhaltet neben der ansprechenden Startseite vor allem eine Präsentation der vorhandenen Fahrzeuge mit den jeweiligen Preisangaben. Und natürlich darf auch ein Kontaktformular für Anfragen nicht fehlen.

1.1.3. Der Verwaltungsbereich

Der Verwaltungsbereich darf nur nach Eingabe von Benutzername und Kennwort zugänglich sein. Da die FahrFlott GmbH nur wenige Mitarbeiter beschäftigt, soll auf eine Benutzerverwaltung verzichtet werden. Stattdessen soll ein fester Zugangsaccount verwendet werden.

Die Verwaltung soll folgende Kriterien erfüllen:

- **Fahrzeugklassen:** Alle vorhandenen Fahrzeuge sind einer Fahrzeugklasse zugeordnet um eine Rechnungserstellung zu vereinfachen. Für jede Fahrzeugklasse lassen sich ein Tagessatz und eine Kilometerpauschale definieren. Außerdem soll ein repräsentatives Bild für jede Fahrzeugklasse verwaltet werden können. Dieses Bild soll auch dem „normalen“ Besucher zugänglich sein.

  Selbstverständlich sollen die vorhandenen Klassen editiert oder gelöscht werden können. Sollten beim Löschen einer Klasse noch Fahrzeuge in dieser Klasse registriert sein, soll ein Hinweis erscheinen und die Klasse nicht gelöscht werden können.

- **Fahrzeuge:** Von jedem Fahrzeug sollen neben der zugewiesenen Fahrzeugklasse noch das amtliche Kennzeichen, der aktuelle Kilometerstand sowie das Baujahr editiert, gelöscht oder neu angelegt werden können.

- **Kunden:** Die Kundenverwaltung soll das Anlegen neuer sowie das Editieren und Löschen vorhandener Kunden ermöglichen. Von den Kunden werden folgende Angaben benötigt: Anrede, Vorname, Nachname, Geburtsdatum, Telefon sowie die Adresse.

  Kunden, die jünger als 18 Jahre sind, dürfen nicht erfasst werden, hier soll gegebenenfalls ein Hinweis erscheinen.

- **Rechnungen:** Für vorhandene Kunden müssen hier die Rechnungen erstellt, geändert und gelöscht werden können. Dabei sollen die Rechnungswerte, also die Beträge für die Mietdauer, die gefahrenen Kilometer und der Gesamtbetrag automatisch berechnet werden. Das Rechnungsdatum ist immer gleich dem Erstelldatum der Rechnung, bei einer nachträglichen Änderung der Rechnungswerte darf es nicht mehr veränderbar sein.

  Der km-Stand des gemieteten Fahrzeugs zu Beginn der Miete wird aus den Fahrzeugdaten als Vorgabewert eingetragen, kann aber nach oben verändert werden. Der Abgabe-km-Stand soll als neuer km-Stand des Fahrzeugs gespeichert werden.

  Außerdem soll die Rechnung als separate Seite zum Druck aufbereitet angezeigt werden können.

1.2. Die Planung

Eine gute Planung zahlt sich spätestens bei Korrekturen und Erweiterungen eines Projektes aus. Bei umfangreichen Projekten ist eine sorgfältige Planung unerläßlich, auch wenn Sie sicher viel lieber gleich mit der Umsetzung anfangen würden.

1.2.1. Einzelseiten, Template oder CMS?

Bei kleinen Websites, die nur einen Visitenkartencharakter haben, können Sie auch heute noch ohne Probleme mit einzeln erstellten Seiten arbeiten. Mit einer guten Entwicklungsumgebung sind auch spätere Änderungen, die viele oder alle der Einzelseiten betreffen, leicht umzusetzen. Der große Vorteil bei Einzelseiten ist meiner Ansicht nach, dass sich jede Seite ganz individuell bearbeiten lässt und sich damit auch ausgefallenere Dinge realisieren lassen. Allerdings ist hier natürlich die große Gefahr von Inkonsistenzen gegeben, vor allem bei Layout- und Designanpassungen.

Auch wenn es heute schick ist, ein CMS (Content Management System) für eine Website einzusetzen, ist das oft mit vielen Nachteilen verbunden. Falls Sie nämlich kein CMS kennen und beherrschen, kommt neben der eigentlichen Website-Erstellung noch der Aufwand des Lernens für das CMS hinzu. Und je nach CMS kann das ganz schön viel Aufwand sein. Und außerdem, welches CMS wollen Sie denn nehmen? Das „Ich kann wirklich alles Monster Typo3“, oder doch lieber den Tausendsassa Joomla? Oder oder oder?

Die englische Wikipedia[1] spuckt eine solche Menge verschiedener CMS aus, alleine um das am besten für Ihr Projekt passende zu finden können Sie schon Tage mit Recherchen zubringen.

Und wenn das gewünschte CMS die benötigte Funktionalität nicht bereitstellt und sie sich auch nicht als Plugin nachrüsten lässt, dann müssten Sie sich die Funktionalität auch noch selber programmieren. In der „Sprache“ des jeweiligen CMS. Und das kann ganz schön aufwendig und lernintensiv sein.

Andererseits, haben Sie sich erst einmal mit einem CMS vertraut gemacht und vielleicht schon die eine oder andere Erweiterung geschrieben, werden Sie wohl nie wieder etwas anderes einsetzen wollen. Ein gutes CMS nimmt Ihnen soviel von der eintönigen Arbeit bei der Seitenerstellung ab und bietet Ihnen so viele Vorteile, dass Sie vermutlich nur mit Grauen an Ihre früheren CMS-losen Projekte zurückdenken. Allerdings benötigen Sie eine ganze Menge an Kenntnissen in HTML, CSS, Javascript, PHP usw., wenn Sie „Ihr“ CMS individuell anpassen wollen. Und genau diese Themen behandeln wir ja in diesem Buch.

Für unser Projekt, und für Sie, die Sie ja schließlich HTML, CSS, PHP usw. kennenlernen möchten, ist die template-basierte Variante meiner Ansicht nach das vernünftigste. Durch den Einsatz eines Templates (Vorlage) reduzieren wir viele der unnötigen und fehleranfälligen Tätigkeiten, die bei Einzelseiten entstehen können, und nutzen

[1] `http://en.wikipedia.org/wiki/List_of_content_management_systems`

gleichzeitig alle Vorteile, die Einzelseiten mit sich bringen. Außerdem lassen sich mit solchen Lösungen durchaus auch größere und komplexe Websites erstellen, ohne dass Sie ein CMS nutzen müssten.

1.2.2. Das Template

Bei diesem Projekt lässt sich die Struktur für die Template-Basisseite wie in Abbildung 1.2 realisieren. Der grundsätzliche Aufbau ist leicht zu durchschauen, es gibt nur vier wesentliche Bereiche.

Kopf- und Fußbereich sind einfach zu realisieren und erhalten mit den zugehörigen IDs eindeutige Kennungen.

Die beiden grau dargestellten Bereiche hätte man früher eventuell über eine Tabellenkonstruktion gelöst, dass lässt sich heute mit CSS jedoch wesentlich besser realisieren. Auch diese beiden Bereiche erhalten eindeutige Kennungen über ihre jeweiligen IDs. Dabei entspricht die ID des rechten Bereichs dem jeweiligen Seitennamen, so dass wir die verschiedenen Seiten gezielt ansprechen können. Zusätzlich weisen wir ihm noch die Klasse '*inhalt*' zu, die wir für die allgemeine Formatierung des Bereichs nutzen werden.

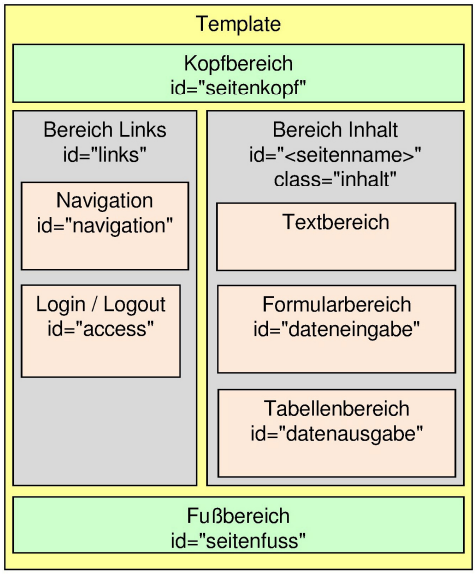

Abb. 1.2.: Das Basislayout

Der linke Bereich enthält dabei zwei Module, die Navigation (also das Menü) und das Modul zur Anmeldung bzw. Abmeldung. Auch hier geben wir beiden wieder eindeutige IDs, um sie später leichter ansprechen zu können.

Im rechten Bereich befinden sich drei unterschiedliche Module. Je nachdem, welche Seite angezeigt wird, werden nicht immer alle drei angezeigt. Auf der Startseite beispielsweise wird nur ein Textbereich nötig sein, auf den Verwaltungsseiten benötigen wir jedoch sowohl Formulare zur Bearbeitung als auch Tabellen zur Anzeige der Daten. Um diese beiden komplexeren Bereiche leichter ansprechen zu können, erhalten sie ebenfalls eindeutige IDs.

1.3. Einführung in Datenbanken

Sämtliche Nutzdaten unserer Website werden in einer Datenbank gespeichert. Falls Sie noch nicht mit Datenbanken gearbeitet haben, es ist wirklich nicht kompliziert.

Stellen Sie sich eine Datenbank einfach als eine Sammlung zusammengehöriger Tabellen vor. Jede dieser Tabellen hat einen eindeutigen Namen und enthält Spalten, die die benötigten Daten aufnehmen. Dabei hat jede dieser Spalten ebenfalls einen eindeutigen Namen. Außerdem ist für jede Spalte noch festgelegt, welche Art von Daten (der Datentyp[2]) in ihr abgelegt werden kann, also beispielsweise Texte, Zahlen oder Datumswerte.

Das Ganze ist also nicht viel anders als in einer Tabellenkalkulation, nur dass dort die Spalten keinen Namen haben müssen und sich in jeder Spalte auch Daten verschiedener Datentypen ablegen lassen.

Für die Arbeit mit Datenbanken ist es äußerst hilfreich, wenn jede Tabelle eine Spalte enthält, die eindeutige Werte beinhaltet, die sogenannten Primärschlüssel. Das erleichtert den Zugriff auf die benötigten Daten enorm und ermöglicht Verbindungen zu anderen Tabellen. Denken Sie hier einmal an Ihre Strom- oder Telefonrechnung. Jede Rechnung hat viele Gemeinsamkeiten mit den anderen Rechnungen, die Anschrift, die Kundennummer, den Absender usw. Aber jede Rechnung hat auch einen eindeutigen Inhalt, nämlich die Rechnungsnummer und die Rechnungsposten.

Einer der großen Vorteile einer Datenbank besteht nun darin, Tabellen anhand dieser eindeutigen Spalten miteinander zu verknüpfen. Bleiben wir einmal bei dem Beispiel Ihrer Stromrechnung. Ihr Energieversorger hat sicherlich eine Tabelle, welche Ihre Kundendaten enthält, also Name, Anschrift, Tarif etc. Und natürlich Ihre eindeutige Kundennummer.

Wenn jetzt eine Rechnung erstellt wird, stehen die Verbrauchsdaten in der Rechnungstabelle. Aber es wäre doch unsinnig, hier auch noch die ganzen Kundendaten mit hineinzuschreiben, denn diese liegen ja schon in der Kundentabelle. Stattdessen steht in der Rechnungstabelle nur Ihre Kundennummer, und alle benötigten Informationen lassen sich dann über diese eindeutige Nummer aus der Kundentabelle abrufen. Damit spart man eine große Menge an Redundanzen (mehrfach gespeicherte Daten) ein, denn Sie und viele andere Kunden erhalten ja viele Stromrechnungen über die Jahre.

Auch bei den Tabellen der FahrFlott-Datenbank werden solche Verknüpfungen auftreten, doch mehr dazu im nächsten Abschnitt.

1.3.1. Die FahrFlott-Datenbank

Wir benötigen für unser Projekt in der Datenbank Tabellen für die Fahrzeugklassen, die Fahrzeuge, die Kunden und die Rechnungen.

Für jede dieser Tabellen müssen wir uns überlegen, welche Daten wir in der Tabelle ablegen möchten, welche Spalten diese Tabelle also enthalten soll und von welchem

[2]Datenbanksysteme unterstützen eine große Menge unterschiedlicher Datentypen, um die Daten optimal speichern und verarbeiten zu können. Wir verwenden hier nur die folgenden Typen: *integer* für ganze Zahlen, *float* für Kommazahlen, *text* für Texte, *date* für Datumswerte und *boolean* für ja/nein Werte

Datentyp die Daten sind. Die konkrete Umsetzung werden wir dann in Kapitel 6 behandeln.

Die Tabelle *klassen* für die Fahrzeugklassen beinhaltet nur wenige Spalten. Die erste Spalte *id_klasse* enthält den Primärschlüssel (ID) der einzelnen Datensätze, um diesen müssen wir uns nicht weiter kümmern, da dessen Verwaltung die Datenbank übernimmt.

In die Spalten *klasse*, *tagessatz* und *kmsatz* schreiben wir dann später die Informationen zur Fahrzeugklasse hinein, also beispielsweise „Limousine“, „60“ und „1.25“. Wichtig ist hier, dass wir keine Einheiten angeben, also nicht etwa „60,-EUR“, denn diese Spalten dürfen nur Zahlen enthalten. Die passende Einheit werden wir später auf der jeweiligen Seite mit PHP oder Javascript hinzufügen.

Die Spalte *bildname* wird den Dateinamen des Bildes für die Fahrzeugklasse enthalten, dieser ist natürlich vom Datentyp *text* und kann somit beliebige Zeichenfolgen aufnehmen. Dennoch sollten wir darauf achten, bei Dateinamen keine Sonderzeichen oder gar Leerzeichen zu verwenden, da viele Serverbetriebssysteme und dort eingesetzte Programmiersprachen, wie etwa auch PHP, damit Probleme bekommen können. Von daher ist es sinnvoll, sich auf die Buchstaben, Ziffern und eventuell den Unterstrich _ zu beschränken.

Die letzte Spalte dient dazu, den Status der jeweiligen Fahrzeugklasse aufzunehmen, also ob sie gelöscht ist oder nicht. Natürlich könnten wir den entsprechenden Datensatz beim Löschen auch aus der Tabelle entfernen, meine Erfahrung hat mir aber gezeigt, dass es oft Fälle gibt, bei denen man gelöschte Daten gerne wiederherstellen möchte, ohne sie komplett neu einzugeben. Und da unsere kleine Datenbank sicher keine Millionen von Datensätzen enthalten wird, haben wir auch keine Probleme, was den Speicherplatz angeht.

Die Tabelle 1.2 *fahrzeuge* für die einzelnen Fahrzeuge sollten Sie jetzt schon fast vollständig selber verstehen, das einzig Neue ist die zweite Spalte *id_klasse*. Hiermit verknüpfen wir jedes Fahrzeug mit einer Fahrzeugklasse und können so für jedes Fahrzeug leicht die benötigten Daten aus der Tabelle *klasse* abrufen.

Zur Tabelle *kunden* muss ich jetzt nichts mehr schreiben, hier tauchen schon keine unbekannten Dinge mehr auf.

Als letztes haben wir noch die Rechnungstabelle, auch hier sollten die einzelnen

Spalte	**Datentyp**	**Hinweis**
id_klasse	integer	Eindeutige ganzzahlige Kennung
klasse	text	Klassenname, beliebiger Inhalt
tagessatz	float	Tagessatz als Kommazahl
kmsatz	float	km-Satz als Kommazahl
bildname	text	Name der Bilddatei, falls vorhanden
geloescht	boolean	ja/nein Zustand

Tab. 1.1.: Die Tabelle *klassen*

Spalte	Datentyp	Hinweis
id_fahrzeug	integer	Eindeutige ganzzahlige Kennung
id_klasse	integer	Verknüpfung zur Fahrzeugklasse
kennzeichen	text	Kennzeichen
baujahr	integer	Baujahr
kmstand	integer	aktueller km-Stand
geloescht	boolean	ja/nein Zustand

Tab. 1.2.: Die Tabelle *fahrzeuge*

Spalte	Datentyp	Hinweis
id_kunde	integer	Eindeutige ganzzahlige Kennung
nachname	text	
vorname	text	
gebdatum	date	Geburtsdatum
strasse	text	
plz	integer	
ort	text	
telefon	text	Textformat wegen Sonderzeichen
geloescht	boolean	ja/nein Zustand

Tab. 1.3.: Die Tabelle *kunden*

Spalte	Datentyp	Hinweis
id_rechnung	integer	Eindeutige ganzzahlige Kennung
id_kunde	integer	Verknüpfung zum Kunden
id_fahrzeug	integer	Verknüpfung zum Fahrzeug
datum	date	Rechnungsdatum
datum_von	date	Miete Startdatum
datum_bis	date	Miete Endedatum
km_von	integer	Mietbeginn km-Stand
km_bis	integer	Mietende km-Stand
geloescht	boolean	ja/nein Zustand

Tab. 1.4.: Die Tabelle *rechnungen*

Spalten für Sie mittlerweile verständlich sein. Ebenso wie in der Tabelle *fahrzeuge* bringen wir hier Verknüpfungen mit anderen Tabellen über die jeweiligen IDs unter.

Solche Verknüpfungen nennt man auch Fremd- oder Sekundärschlüssel, da sie sich auf Daten einer anderen Tabelle beziehen. Diese Verknüpfungen sind für die Rechnungstabelle in Abbildung 1.3 verdeutlicht.

Bei unserer Rechnungstabelle handelt es sich um eine minimale Form, da wir hier auf alle Daten verzichten, die sich aus bereits vorhandenen Werten berechnen lassen. So benötigen wir weder eine Spalte für die Anzahl der Tage noch eine Spalte für den Rechnungsbetrag.

Die Anzahl der Tage können wir jederzeit über die Werte der Spalten *datum_von* und *datum_bis* berechnen, die gefahrenen Kilometer über die Werte *km_von* und *km_bis*. Über die ID des Fahrzeugs können wir die Fahrzeugklasse herausbekommen, und damit die Werte für den Tagessatz und den km-Satz. Und so können wir dann auch den Rechnungsbetrag bestimmen.

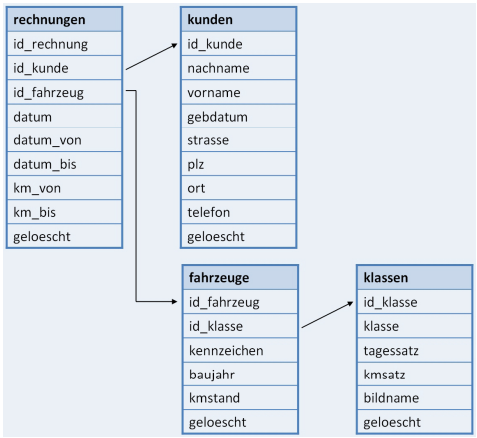

Abb. 1.3.: Die Verknüpfungen der Rechnungstabelle

Ideal ist das allerdings nicht, denn wenn die FahrFlott GmbH einmal die Werte für eine Klasse ändert, erhalten wir andere Rechnungswerte als bei der ursprünglichen Rechnung. Von daher wäre es durchaus sinnvoll, die Tabelle um die klassenspezifischen Spalten *tagessatz* und *kmsatz* zu erweitern. Aber das können Sie ja später leicht als Übung für die Rechnungsverwaltung selber durchführen. Oder Sie schauen im Abschnitt 12.1 nach.

2. Struktur mit HTML

2.1. Was ist eigentlich HTML?

HTML, die *Hyper Text Markup Language*, ist eine Seitenbeschreibungssprache. Das bedeutet, in dieser Sprache gibt es Elemente, die die strukturelle Gliederung der jeweiligen Seiteninhalte beschreiben.

Anders als zum Beispiel in einer Textverarbeitung, wo Sie die Inhalte (Überschriften, Absätze, Tabellen etc.) nach Ihren Wünschen formatieren können, geben Sie in HTML neben den reinen Inhalten auch die Art der Inhalte an.

In der Textverarbeitung würden Sie vielleicht Überschriften in der Schriftart Times mit 17pt Größe in fett formatieren, für das Textverarbeitungsprogramm handelt es sich dabei aber immer noch um ganz normalen Text mit einer bestimmten Formatierung.

In HTML schreiben Sie jedoch in das Dokument zusätzlich zum reinen, unformatierten Text noch die Information, dass es sich hierbei um eine Überschrift handelt. Der Browser erkennt dann diese Information und formatiert die Überschrift anhand seiner Voreinstellungen oder anhand der Vorgaben in der zugehörigen CSS-Datei.

Um ein HTML-Dokument zu erstellen, müssen Sie also neben den reinen textlichen Inhalten auch die benötigten strukturellen Informationen mit angeben. Die dazu verwendeten Befehle nennt man *tags*.

Während es früher nur wenige essentielle tags gab, versucht man heute mit der fünften Version von HTML weitere semantische Informationen im Dokument unterzubringen. Auf diese Weise kann zum Beispiel das Vorleseprogramm eines sehbehinderten Website-Besuchers nicht relevante Teile, wie etwa den Seitenkopf oder die Fußzeile, erkennen und nur den inhaltlich wichtigen Teil der Seite vorlesen.

Außerdem ist es ein wichtiges Ziel, die Strukturierung und den eigentlichen Inhalt einer Seite von den Designangaben zu trennen. Dadurch erreicht man nicht nur leichter lesbare Dokumente, sondern kann spätere Änderungen am Design wesentlich einfacher durchführen oder Dokumente für unterschiedliche Ausgabeformate aufbereiten. Wie das mit der optischen Formatierung vor sich geht, werden wir dann beim Thema CSS in Kapitel 4 besprechen.

HTML hat sich seit seiner Erfindung 1992 stark weiterentwickelt und immer wieder Änderungen in der formalen Syntax der Schreibweise erfahren. Aus diesem Grund sind die meisten Webbrowser sehr tolerant, was Fehler bei den tags angeht. Im weit verbreiteten Internet-Explorer von Microsoft gibt es dazu sogar einen speziellen „Kompatibilitätsmodus“. Nichtsdestotrotz sollten Sie natürlich versuchen, syntaktisch kor-

rekten Code zu schreiben, auch wenn die Seite in Ihrem Browser trotz eventueller Fehler richtig angezeigt wird.

Die wichtigste syntaktische Regel bei HTML lautet:

> Zu jedem öffnenden tag gehört auch ein schließender tag

Die einzelnen tags werden dabei von den „Kleiner“ < und „Größer“ > Zeichen umschlossen. Das schließende tag bekommt dabei noch einen Schrägstrich / vorangestellt. Allgemein sieht das Ganze dann so aus:

```
<tag> Inhalt </tag>
```

Es gibt allerdings einige Ausnahmen von dieser Regel, etwa für tags, die keinen Inhalt umschließen. Diese tags enthalten den abschließenden Schrägstrich bereits im öffnenden tag selber. Solche tags haben dann diese Struktur:

```
<tag />
```

2.2. Der Aufbau einer Seite

Ganz am Anfang einer HTML-Seite befindet sich das tag *doctype*. Es kennzeichnet die Dokumentendeklaration DTD der Seite und wird vom Browser ausgewertet. In seinen Attributen finden sich Angaben darüber, welcher HTML-Standard verwendet wird. Bis HTML Version 4 war diese Angabe ausgesprochen kompliziert, das ist mit HTML5 deutlich besser geworden. Eine korrekte Deklaration für eine HTML5-Seite sieht so aus:

```
<!DOCTYPE html>
```

Das Besondere am *doctype* ist, dass es kein schließendes tag hierfür gibt.

Das darauf folgende tag kennzeichnet den Beginn der HTML-Seite und heißt schlicht *html*. Optional kann man noch eine Sprachinformation über das Attribut *lang* hinzufügen, die vom Browser ausgewertet wird.

Das *html*-tag umschließt die beiden Bereiche, aus denen eine HTML-Seite besteht, den Kopf (*head*) und den Körper (*body*). Dabei hat der Kopf nichts mit der Ausgabe vom Seitenkopf zu tun, er beinhaltet vielmehr weitere, vor allem technische Informationen über das Dokument. Das können Angaben zum Autor des Dokumentes, dem Seitentitel der im Browser angezeigt wird oder Informationen über einzubindende Dateien sein. Die eigentlichen strukturellen tags für die Webseite stehen im Körper. Damit sieht das Grundgerüst einer Webseite wie in Abbildung 2.1 aus.

In diesem kleinen Listing finden Sie in den Zeilen 4 und 7 auch eine weitere interessante Information, nämlich wie Sie Kommentare in Ihrem HTML-Code unterbringen können. Solche Kommentare sind nicht nur während der Entwicklung einer Website sinnvoll, vor allem beim späteren Bearbeiten können Sie Ihnen viel Zeit sparen. Deshalb rate ich Ihnen, ruhig umfangreich zu kommentieren, es macht sich bezahlt.

```
<!DOCTYPE html>
<html lang="de">
  <head>
    <!-- Hier stehen Angaben für den Kopfbereich -->
  </head>
  <body>
    <!-- Hier stehen Angaben für den Körper -->
  </body>
</html>
```

Abb. 2.1.: Das Grundgerüst einer Webseite

2.3. Die wichtigsten tags im Kopfbereich

Schauen wir uns nun einmal die tags an, die Sie im Kopfbereich der Seite verwenden können.

2.3.1. Der Seitentitel

Das *title*-tag dient dazu, dem Browser mitzuteilen, wie der Seitentitel ist. Diese Information wird in der Titelleiste bzw. im Tab des Browsers angezeigt und sollte deshalb immer einen sinnvollen Bezug zur angezeigten Webseite haben.

```
<title> FahrFlott GmbH </title>
```

2.3.2. Die Meta-Angaben

In den *meta*-tags können Sie weitere Informationen über Ihre Webseite unterbringen, die etwa von Suchmaschinen ausgewertet werden. Aber auch Angaben über technische Aspekte Ihrer Webseite sind hier möglich. Beachten Sie, dass die *meta*-tags zu den wenigen Ausnahmen gehören, die kein schließendes tag erfordern.

In der ersten Zeile von Abbildung 2.2 findet sich gleich eine recht technische Angabe, nämlich über die Art des Dokuments *text/html* sowie die verwendete Zeichenkodierung *utf-8*. Auch wenn die Browser meist die Zeichenkodierung einer Webseite sehr

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<meta name="description" content="Der Fahrzeugverleih FahrFlott GmbH
    bietet Ihnen zu fairen Preisen tolle Angebote in verschiedenen
    Fahrzeugklassen."/>
<meta name="author" content="Jörg Barres"/>
<meta name="keywords" content="PKW, Oldtimer, Sportwagen, Verleih,
    Miete"/>
<meta name="date" content="2014-03-15"/>
```

Abb. 2.2.: Einige *meta*-tags

zuverlässig ermitteln können, ist es auf keinen Fall falsch, diese Angabe zu machen. Sicher haben Sie schon einmal Webseiten besucht, bei denen manche Zeichen nicht korrekt angezeigt wurden.

Die meisten *meta*-tags informieren über das Attribut *name* darüber, was sie beschreiben. Der zugehörige Inhalt wird dann mit *content* beschrieben.

In Zeile zwei befindet sich eine kurze Beschreibung (*description*) des Angebots der Webseite. Dieser Text wird von manchen Suchmaschinen bei einem Treffer angezeigt und sollte deshalb kurz und prägnant sein.

Zeile drei enthält eine Angabe über den Autor (*author*) der Webseite, diese Angabe ist für Internet-Seiten weniger interessant, bei einer Intranet-Anwendung können hierüber aber leichter Seiten von bestimmten Personen gefunden werden.

Die vierte Angabe enthält Schlüsselwörter (*keywords*) über den Inhalt der Seite. Diese Angabe war früher essentiell für eine gute Indizierung bei Suchmaschinen, wird aber heute kaum noch beim Ranking gewichtet.

In Zeile fünf schließlich steht das Erstellungsdatum (*date*) der Seite. Diese Angabe hilft Suchmaschinen dabei, festzustellen, ob sich die Seite seit dem letzten Besuch des Suchmaschinen-Indizierers geändert hat.

2.3.3. Einbinden von Programmbibliotheken

Das *script*-tag dient dazu, Programmbibliotheken in die Webseite einzubinden. Bei modernen Webseiten handelt es sich dabei in aller Regel um Javascript-Dateien. Neben der Angabe zur Quelle der Datei (*src*) sollte optional auch der Typ (*type*) der Datei mit der Angabe der Programmiersprache sowie die verwendete Zeichenkodierung (*charset*) angegeben werden.

```
<script type="text/javascript" charset="utf-8" src="file.js"> </script>
```

2.3.4. Einbinden von zugehörigen Dateien

Über das *link*-tag lassen sich zugehörige Dateien einbinden. Dabei handelt es sich allerdings nicht um ausführbare Dateien wie beim *script*-tag, sondern um Dateien, die einen logischen Bezug zur Webseite haben, wie etwa CSS-Dateien. Die Art des Bezugs gibt man mit dem Attribut *rel* an, die Quelle der Datei über das Attribut *href* und den Typ mit dem Attribut *type*.

```
<link rel="stylesheet" href="styles.css" type="text/css" />
```

2.4. Die wichtigsten tags im Körper

Kommen wir jetzt endlich zu den tags, die Ihre Website mit Inhalt füllen werden. Ich werde Ihnen diese tags hier nur kurz vorstellen, damit Sie einen Überblick bekommen,

die reale Nutzung besprechen wir, wenn wir die tags dann auf den verschiedenen Seiten einsetzen werden.

Beachten Sie bitte auch, dass die tags vorerst mit der vom Browser vorgegebenen Standardformatierung verwendet werden, die eigentliche optische Formatierung und Ausrichtung erfolgt erst später mit den CSS-Angaben. Damit erreichen wir dann auch die gewünschte strikte Trennung von Inhalt und Design.

2.4.1. Container

Container sind eine interessante Sache, denn sie können praktisch alles enthalten, Texte, Bilder, Tabellen und Multimedia-Elemente. Bis zur HTML-Version 4 gab es nur zwei Container, *span* und *div*.

Der *span*-Container dient vor allem dazu, einen kleinen Textbereich zu umschließen, um etwa eine Hervorhebung innerhalb einer Zeile zu ermöglichen.

Der *div*-Container ist als universeller Container konzipiert und dient dazu, Bereiche der Seite zu strukturieren.

Der größte Unterschied zwischen den beiden ist, dass der *div*-Container einen Block über die gesamte Breite darstellt, also gewissermaßen einen Zeilenumbruch am Ende durchführt, der *span*-Container jedoch nicht.

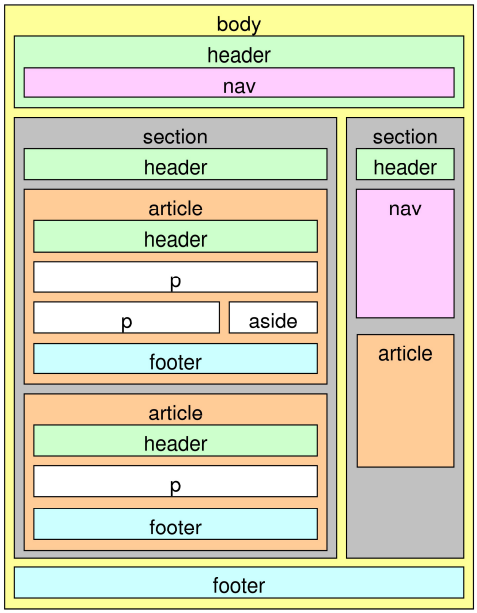

Abb. 2.3.: Semantische Struktur

Diese beiden Container ermöglichen zwar einen strukturierten Aufbau der einzelnen Bereiche einer Webseite, oft ist es aber unumgänglich, viele solche Container ineinander zu verschachteln. Um die Container dennoch gezielt ansprechen zu können, hat man ihnen oft über das Attribut *id* eine eindeutige Bezeichnung gegeben.

Sehen Sie sich doch jetzt einmal Abbildung 2.3 an. Deutlich ist die semantische Strukturierung der Seite zu erkennen.

Neben dem Kopf- und Fußbereich befinden sich zwei nebeneinanderliegende Sektionen auf der Seite. In der linken Sektion, dem Hauptbereich dieser Seite, befinden sich außer dem Kopf der Sektion zwei Artikelbereiche, die beide über eigene Kopf- und Fußbereiche verfügen. Neben dem unteren Absatz des oberen Artikels ist auch ein Bereich für Zusatzinformationen (*aside*) vorgesehen. Die rechte Sektion enthält außer dem Kopfbereich noch einen Navigationsbereich und einen Artikelbereich, etwa für aktuelle Nachrichten. Den Code für diese Anordnung finden Sie in Abbildung 2.4.

```
1  <body>
2    <header>      <!-- Kopfbereich Seite        -->
3      <nav>       <!-- Navigationselemente      --> </nav>
4    </header>
5    <section>     <!-- Sektion links            -->
6      <header>    <!-- Kopfbereich Sektion      --> </header>
7      <article id="oben" >
8        <header> <!-- Artikelheader             --> </header>
9        <p>      <!-- Absatz mit Text           --> </p>
10       <aside>  <!-- Zusatzinformationen       --> </aside>
11       <p>      <!-- Absatz mit Text           --> </p>
12       <footer> <!-- Fußzeile Artikel          --> </footer>
13     </article>
14     <article id="unten" >
15       <header> <!-- Artikelheader             --> </header>
16       <p>      <!-- Absatz mit Text           --> </p>
17       <footer> <!-- Fußzeile Artikel          --> </footer>
18     </article>
19   </section>
20   <section>     <!-- Sektion rechts           -->
21     <header>    <!-- Kopfbereich Sektion      --> </header>
22     <nav>       <!-- Navigationselemente      --> </nav>
23     <article>   <!-- Artikelinhalt            --> </article>
24   </section>
25   <footer>      <!-- Fußbereich Seite         --> </footer>
26 </body>
```

Abb. 2.4.: Container in HTML

Mit der Einführung der neuen Containertypen *header*, *footer*, *nav*, *section*, *article* und *aside* in HTML 5 ist es jetzt möglich, eine Webseite nicht nur strukturiert, sondern auch semantisch korrekt aufzubauen. Um die einzelnen Elemente mit Javascript oder CSS anzusprechen, vergibt man aber auch heute noch eine ID, siehe Zeile sieben und 14 in Abbildung 2.4.

2.4.2. Bilder und links

Eine der wichtigsten Dinge überhaupt im Web sind die links. Ohne sie hätte sich das World Wide Web niemals so entwickelt, wie wir es heute kennen. Ermöglicht es ein link doch, mit einem einzigen Klick zu einer anderen Information zu gelangen. Und diese muss nicht einmal auf dem gleichen Webserver vorhanden sein, oder im gleichen Land.

```
1 <a href="startseite.html"> Zur Startseite </a> <br />
2 <img src="bilder/front1.jpg" alt="FahrFlott-Logo"/>
```

Abb. 2.6.: Bilder und links

In HTML lassen sich links sehr einfach über das tag *a* realisieren. Damit der link aber funktioniert, ist zumindest die Zielangabe für den link über das Attribut *href* notwendig.

Um ein Bild in HTML einzubinden, verwenden wir das tag *img* (für image). Es benötigt zwingend das Attribut *src*, das die Quelle der Bilddatei angibt. Ein weiteres wichtiges, aber optionales Attribut heißt *alt* (alternative).

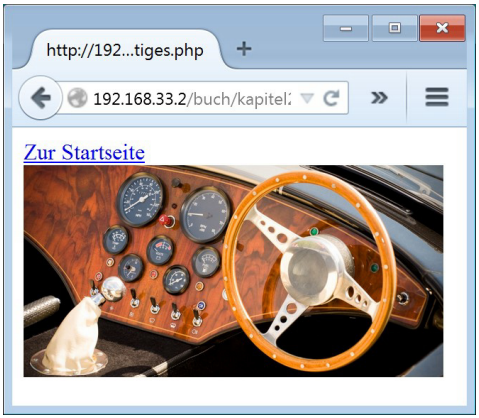

Abb. 2.5.: Bilder und links

Als die Internetverbindungen noch langsam waren, hatten viele Nutzer die Anzeige von Bildern im Browser deaktiviert, da Bilder ja große Datenmengen bedeuten. Und statt eines Bildes hat der Browser dann den Inhalt des *alt*-Attributes angezeigt.

Auch heute noch ist die Angabe des *alt*-Attributes sehr sinnvoll, denn sehbehinderte Menschen können ein Bild nicht erkennen. Deren Vorleseprogramme können aber darauf hinweisen, das hier ein Bild ist, und den Inhalt des *alt*-Attributes vorlesen. Aber auch Suchmaschinen nutzen die Angaben im *alt*-Attribut.

Alle anderen Attribute, wie etwa *height*, *width* oder *border*, sind heute nicht mehr notwendig, da deren Umsetzung über CSS besser gelöst werden kann.

Wenn Sie sich Zeile eins vom Quellcode in Abbildung 2.6 ansehen, finden Sie dort auch noch den *br*-tag. Mit ihm führen Sie einen Zeilenumbruch durch. Der *br*-tag ist eine der wenigen Ausnahmen ohne schließendes tag.

2.4.3. Tabellen

Auch wenn früher Tabellen oft zur Gestaltung einer Webseite eingesetzt wurden, sollten Sie sich darauf beschränken, Tabellen wirklich nur dann zu nutzen, wenn Sie Daten in tabellarischer Form darstellen wollen.

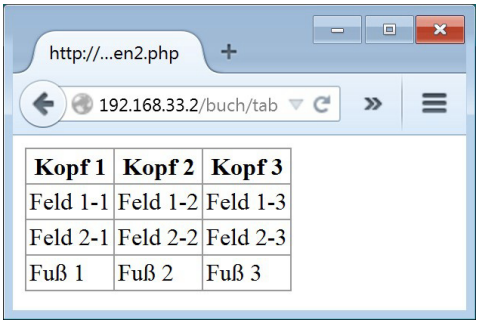

Abb. 2.7.: Eine Tabelle

Der Umgang mit Tabellen ist einfach, jede Tabelle wird durch das tag *table* eingeleitet, dann kommen optionale Bereichsangaben für den Tabellenkopf *thead*, den Tabellenfuß *tfoot* und den Tabelleninhalt *tbody*. Durch diese Einteilung ist es möglich, beim Druck einer Tabelle den Kopf und Fuß auf jeder gedruckten Seite zu wiederholen. Wichtig ist dabei,

```
<table>
  <thead>
    <tr>
      <th> Kopf 1 </th>      <th >Kopf 2 </th>      <th> Kopf 3 </th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td> Fuß 1 </td>       <td> Fuß 2 </td>       <td> Fuß 3 </td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td> Feld 1- 1</td>    <td> Feld 1-2 </td>    <td> Feld 1-3 </td>
    </tr>
    <tr>
      <td> Feld 2-1 </td>    <td> Feld 2-2 </td>    <td> Feld 2-3 </td>
    </tr>
  </tbody>
</table>
```

Abb. 2.8.: Tabellen in HTML

die Reihenfolge einzuhalten, also den *tfoot*-Bereich vor den *tbody*-Bereich zu setzen.

Jede Tabelle besteht aus einer oder mehreren Zeilen, diese werden durch den tag *tr* (table-row) begrenzt. Und jede Zeile wiederum besteht aus der gleichen Anzahl von Feldern, diese werden mit dem tag *td* (table-data) begrenzt. Eine Ausnahme bilden die Felder im Tabellenkopf, hier sollten Sie statt *td* besser *th* (table-head) verwenden.

Es ist natürlich auch möglich, mehrere Felder zusammenzufassen, sowohl vertikal (Attribut *rowspan*) als auch horizontal (Attribut *colspan*).

2.4.4. Aufzählungen

Aufzählungen sind in HTML sehr einfach zu realisieren. Dabei müssen Sie lediglich unterscheiden, ob Sie eine Aufzählung mit einer Nummerierung *ol* (ordered list) oder eine ohne Nummerierung *ul* (unorderd list) haben möchten. Die einzelnen Aufzählungspunkte werden dann mit dem tag *li* (list item) hinzugefügt. Standardmäßig werden Aufzählungen etwas vom Text abgesetzt und eingerückt. Selbstverständlich sind auch verschachtelte Aufzählungen möglich, ebenso lässt sich die Nummerierung oder das Aufzählungszeichen verändern.

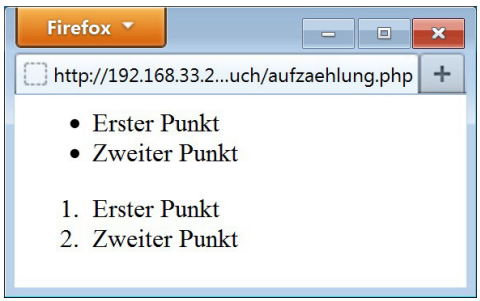

Abb. 2.9.: Aufzählungen

```
<ul>
  <li> Erster Punkt </li>
  <li> Zweiter Punkt </li>
</ul>
<ol>
  <li> Erster Punkt </li>
  <li> Zweiter Punkt </li>
</ol>
```

Abb. 2.10.: Aufzählungen in HTML

2.4.5. Überschriften und Absätze

Überschriften und Absätze dienen dazu, die Inhalte der Seite strukturiert anzuzeigen.

So werden Überschriften standardmäßig größer und fetter als der normale Text dargestellt und erhalten nach oben und unten einen Abstand. Überschriften werden mit den tags *h1* bis *h6* gekennzeichnet. Dabei steht *h1* für eine Überschrift der Ebene eins, also die „wichtigste“ Überschrift.

Textabsätze erhalten einen kleinen Abstand zum folgenden Absatz, so wie Sie es auch aus Büchern oder Zeitschriften kennen. Das tag für Absätze ist *p* (paragraph).

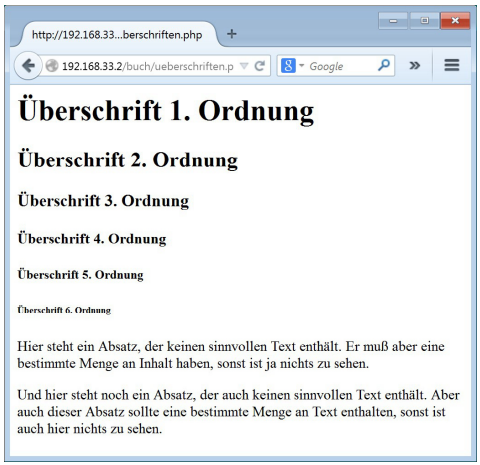

Abb. 2.11.: Überschriften und Absätze

```
<h1> Überschrift 1. Ordnung </h1>
<h2> Überschrift 2. Ordnung </h2>
<h3> Überschrift 3. Ordnung </h3>
<h4> Überschrift 4. Ordnung </h4>
<h5> Überschrift 5. Ordnung </h5>
<h6> Überschrift 6. Ordnung </h6>
<p> Hier steht ein Absatz, der keinen sinnvollen Text enthält. Er muß
    aber eine bestimmte Menge an Inhalt haben, sonst ist ja nichts zu
    sehen. </p>
<p> Und hier steht noch ein Absatz, der auch keinen sinnvollen Text
    enthält. Aber auch dieser Absatz sollte eine bestimmte Menge an
    Text enthalten, sonst ist auch hier nichts zu sehen. </p>
```

Abb. 2.12.: Überschriften und Absätze in HTML

2.5. Das Template der Website - Teil 1

Nachdem Sie jetzt die Grundlagen von HTML gelernt haben, wird es Zeit, das Template für die FahrFlott GmbH zu entwerfen. Anhand des Designentwurfs können wir die Einteilung in die verschiedenen Bereiche wie in Abbildung 2.13 leicht umsetzen.

Wie Sie sehen, verwenden wir viele der neuen semantischen tags, um unsere Seite nach modernen Standards zu gestalten. Damit wir die einzelnen Bereiche später ansprechen können, um sie mit CSS zu formatieren oder um mit Javascript darauf zuzugreifen, setzen wir für die meisten Bereiche IDs ein.

Der rechte Bereich weist dabei zwei Besonderheiten auf. Dort werden wir die ID des Bereichs so setzen, dass sie dem Seitennamen entspricht. Um diesen Bereich dennoch leicht mit CSS formatieren zu können, weisen wir ihm außerdem die Formatierungsklasse „*inhalt*“ zu.

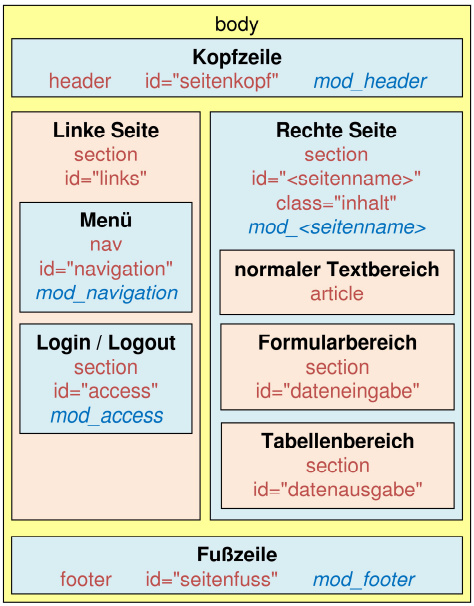

Abb. 2.13.: Das Template

Ich verwende in dieser Abbildung folgende Modalitäten:

- Ein blauer Hintergrund steht für eine Moduldatei, die in das Template geladen wird.
- Ein roter Hintergrund bedeutet, der Code steht direkt in der jeweiligen Datei.
- Der fettgeschriebene Text stellt die umgangssprachliche Bezeichnung des jeweiligen Blockes dar.
- Danach folgen in roter Schrift der verwendete HTML-tag sowie die ID und eventuell die CSS-Klasse.
- Der blaue kursive Text schließlich bezeichnet den Dateinamen des Moduls.

2.5.1. Die Template-Datei *index.php*

Bevor Sie sich jetzt an die Erstellung des Templates machen, müssen Sie noch einige Dinge beachten. Denn da wir bislang das Thema PHP noch nicht bearbeitet haben,

können Sie die einzelnen Module für Kopfbereich, Fußbereich, linke Seite und rechte Seite noch nicht einfügen.

Sie könnten das natürlich so lösen, dass Sie den benötigten HTML-Code, etwa für den Kopf- oder Fußbereich, direkt in Ihre Template-Datei schreiben. Das widerspräche aber dem Sinn eines Templates, das wirklich nur den äußeren Rahmen vorgeben soll.

Erstellen Sie die Template-Datei mit dem Namen *index.php* also erst einmal nur mit den benötigten tags, aber noch ohne Inhalt in den einzelnen Bereichen. Also genau so, wie Sie es bei den Containern in Abbildung 2.4 sehen.

Moment mal, wieso eigentlich *index.php* und nicht *index.html*?

Momentan würde unsere Seite auch mit der Endung *.html* funktionieren, aber da wir später PHP-Code auf dieser Seite ausführen wollen, teilen wir das dem Webserver

```
41 <!DOCTYPE html>
42 <html lang="de">
43 <head>
44   <!-- Meta-tags -->
45   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
46   <meta name="description" content="Der Fahrzeugverleih FahrFlott GmbH
         bietet Ihnen zu fairen Preisen tolle Angebote in verschiedenen
         Fahrzeugklassen."/>
47   <meta name="author" content="Jörg Barres"/>
48   <meta name="keywords" content="PKW, Oldtimer, Sportwagen, Verleih,
         Miete"/>
49   <meta name="date" content="2014-03-15"/>
57   <title> FahrFlott GmbH </title>
58 </head>
60 <body>
61   <header id="seitenkopf">
62     <!-- Modul für den Kopfbereich -->
64   </header>
65   <section id="links">
66     <nav id="navigation">
67       <!-- Modul für die Navigation -->
69     </nav>
70     <section id="access">
71       <!-- Modul für Login / Logout -->
73     </section>
74   </section>
75   <section id="" class="inhalt">
76     <!-- Modul für den Seiteninhalt -->
78   </section>
79   <footer id="seitenfuss">
80     <!-- Modul für den Fußbereich -->
82   </footer>
83 </body>
84 </html>
```

Abb. 2.14.: Das Basistemplate *index.php*

bereits jetzt durch die Endung *.php* mit. Für den aktuellen Inhalt unseres Templates macht das keinen Unterschied.

Beachten Sie bitte, dass Sie im rechten Bereich nur die Sektion selber anlegen müssen, nicht die eingeschlossenen Bereiche *article* oder *dateneingabe* / *datenausgabe*. Diese werden von den jeweiligen Modulen später selber bereitgestellt.

Ergänzen Sie diese Datei dann noch um die wichtigsten Angaben für den *head*-Bereich, also den *title*-tag und einige Meta-Angaben.

Wenn Sie das erledigt haben, sollte Ihre Datei in etwa so aussehen wie der Code in Abbildung 2.14. Die angezeigten Zeilennummern beziehen sich bereits auf die fertige Datei *index.php* um Ihnen spätere Änderungen leichter zu machen. Momentan können Sie die Nummern aber noch ignorieren.

Öffnen Sie jetzt einmal die Seite in Ihrem Browser. Aber Achtung, nicht über einen Doppelklick auf den Dateinamen, sondern über die richtige URL der Seite, also in der Form `http://server-ip/ordner/index.php`.

Wenn alles klappt, sehen Sie eine weiße Seite. Wir haben ja bislang außer der Struktur noch keine Inhalte eingegeben und auch noch keine CSS-Datei zur Formatierung angelegt. Aber ganz oben, in der Titelleiste des Browsers oder des Tabs, da sollte der Titel der Webseite zu sehen sein. Vielen Dank an das *title*-tag hierfür.

2.5.2. Der Firebug im Firefox

Das ist jetzt ein guter Moment, um einen ersten Blick auf ein nahezu unverzichtbares Werkzeug bei der Webentwicklung zu werfen, den Firebug. Dieses Add-on für den Firefox-Webbrowser stellt nämlich eine große Menge an hilfreichen Tools bereit, um Fehler auf Webseiten zu finden, Anpassungen zu machen oder den Code zu optimieren.

Mittlerweile hat zwar praktisch jeder gängige Browser solche Hilfsmittel auch fest eingebaut, beim Internet-Explorer und Chrome heißen Sie Entwicklertools und beim Firefox Inspektor. Den Firefox samt Firebug gibt es aber für jede Betriebssystem-Plattform, es spielt also keine Rolle, ob Sie unter Windows, OS X oder Linux arbeiten. Aber Sie können natürlich trotzdem gerne andere Hilfsmittel nutzen, falls Sie bereits mit einem speziellen System vertraut sind.

Wenn Sie jetzt also diese weiße Seite im Firefox sehen, machen Sie einmal einen Rechtsklick an eine beliebige Stelle auf der Seite. Im erscheinenden Kontextmenü wählen Sie den Befehl „*Element mit Firebug untersuchen*“. Es öffnet sich ein zweigeteilter Bereich am unteren Fensterrand, der Ihnen links den HTML-Code und rechts den CSS-Code zeigt. In Abbildung 2.15 ist der CSS-Code aus Platzgründen ausgeblendet, außerdem steht ja sowieso noch nichts drin.

Im linken Bereich können Sie mit einem Klick auf das kleine + die Hierarchie öffnen und mit einem Klick auf das kleine - wieder schließen. Öffnen Sie einmal den *head*-Bereich. Dort sollten Sie Ihre *meta*-tags und das *title*-tag sehen.

Wenn Sie jetzt noch die *body*-Hierarchie öffnen, sehen Sie, dass die Webseite wirklich Ihren Code enthält.

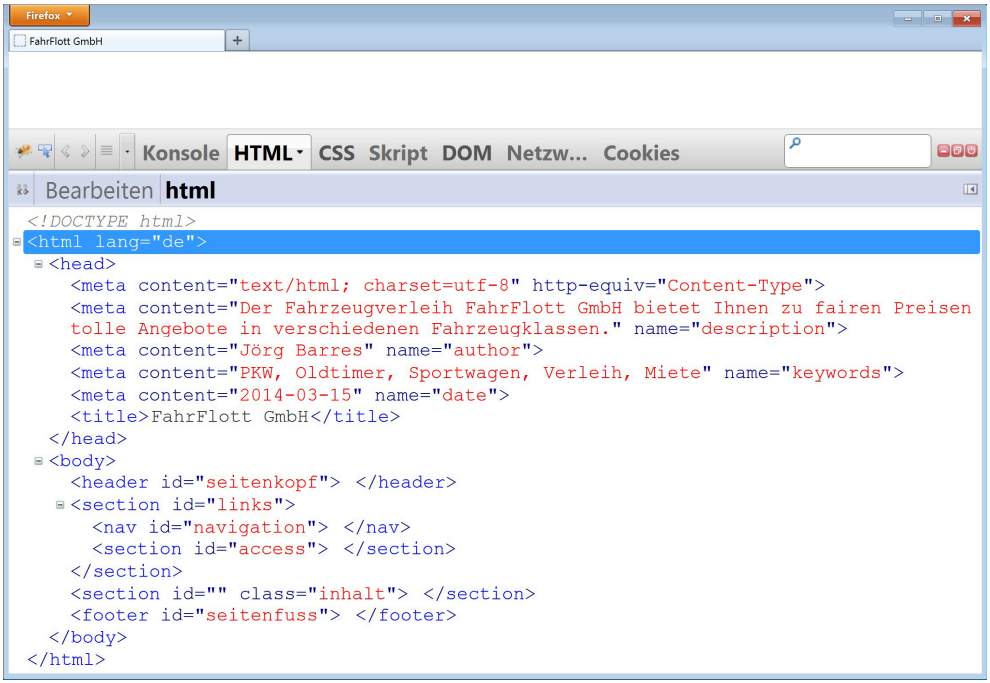

Abb. 2.15.: Das Analysetool Firebug

Index