3 XML-Grundlagen

In diesem Kapitel erfahren Sie, wie Sie einfache XML-Dokumente mit Elementen erstellen können, die innerhalb dieser Dokumente von Bedeutung sind. Sie werden in den Umgang mit Stylesheets eingeführt und Ihre ersten kleinen XML-Dokumente schreiben. Diese können Sie dann mit Hilfe eines Webbrowsers anschauen.

3.1 Hello World!

Es ist Tradition, dass das erste Programm einer neuen Programmiersprache den Schriftzug `Hello World!` auf dem Bildschirm ausgibt. Fangen wir also mit diesem Beispiel an.

Starten Sie bitte Ihren Editor und geben Sie die folgenden Zeilen ein:

```
<?xml version="1.0" standalone="yes"?>
<!-- Hier beginnen die XML-Daten -->
<A00>
Hello World!
</A00>
```

Sehen wir uns die einzelnen Zeilen mal genauer an:

Die erste Zeile ist die XML-Deklaration: `<?xml version="1.0" standanlone="yes"?>`. An der Deklaration erkennt man, dass es sich um ein XML-Dokument handelt. Sie wird immer an den Anfang einer XML-Datei gesetzt; in diesem Fall enthält sie zum einen die Informationen, dass das Dokument mit der Version 1.0 des XML-Standards übereinstimmen sollte, zum anderen wird mit dem Attribut `standalone="yes"` angezeigt, dass alle zur Darstellung benötigten Daten bereits in der Datei vorhanden sind und kein Import von weiteren Dateien erforderlich ist.

Die darauffolgende Zeile ist ein Kommentar und dient der Verständlichkeit der Quelltexte. In XML beginnen und enden Kommentare immer mit `<!--` und `-->`. Was innerhalb dieser Klammern steht, wird vom Browser ignoriert. Die Kommentare dürfen weder vor der XML-Deklaration noch innerhalb eines XML-Element-Tags auftauchen.

Die nächsten drei Zeilen bilden zusammen das Element `A00`.

```
<A00>
Hello World!
</A00>
```

Ein XML-Element besteht aus einem Start-Tag und einem End-Tag, in diesem Fall `<A00>` und `</A00>`. Diese beiden umschließen den Inhalt des Elements `A00`.

Die Bedeutung des Elements `A00` ist im Augenblick noch nicht definiert. XML ist im Gegensatz zu HTML nicht auf vordefinierte Tags angewiesen. In XML kann man seine eigenen Tags erstellen und ihnen dann die Bedeutung zuweisen, die man für das Element vorgesehen hat. Man hätte dem Element auch einen anderen Namen, zum Beispiel `<P>`, geben können, durch das Zuweisen der Bedeutung übernimmt es dann die gleiche Aufgabe. Ein Webbrowser, der nur HTML versteht, würde bei dem Tag `<P>` und `</P>` die Bedeutung `Absatz` zuweisen, der Computer folgt einfach den Regeln, die vorher festgelegt worden sind, und es ist egal, ob der Computer bei dem Tag `<A00>` oder bei dem Tag `<P>` vor dem nächsten Element eine Leerzeile darstellt. Seine Bedeutung, wie es also auf dem Bildschirm ausgegeben wird, erhält ein Tag mit Hilfe von Stylesheets.

Sehen wir uns nun zunächst das Ergebnis an. Damit Sie Ihr Dokument ansehen können, müssen Sie den Quelltext auf Ihrer Festplatte abspeichern. Dazu legen Sie bitte auf Ihrer Festplatte ein Verzeichnis an, in dem Sie dieses und auch Ihre zukünftigen XML-Dateien ablegen können. Geben Sie nun Ihrem Dokument einen Namen Ihrer Wahl (z.B.: `HelloWorld.xml`), wobei Sie darauf achten müssen, die Namenserweiterung `.xml` zu benutzen.

Hinweis

Bitte beachten Sie auch, dass der Quelltext im Textformat abgespeichert werden muss. Wenn Sie ein Dokument zum Beispiel im Word-Format abspeichern, werden zusätzliche Steuerzeichen abgespeichert, die eine XML-Datei unbrauchbar machen.

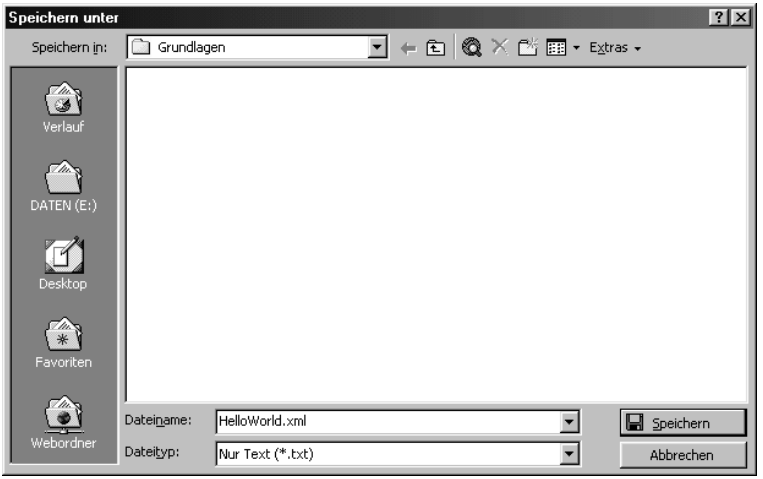

Bild 3.1: XML-Dateien sind im Textformat abzuspeichern.

Nachdem Sie nun Ihre erste XML-Datei erstellt haben, wollen wir uns das Ergebnis ansehen. Dazu benötigen wir einen XML-fähigen Webbrowser, zum Beispiel den Internet Explorer 5.0. Das Ergebnis unterscheidet sich von Browser zu Browser.

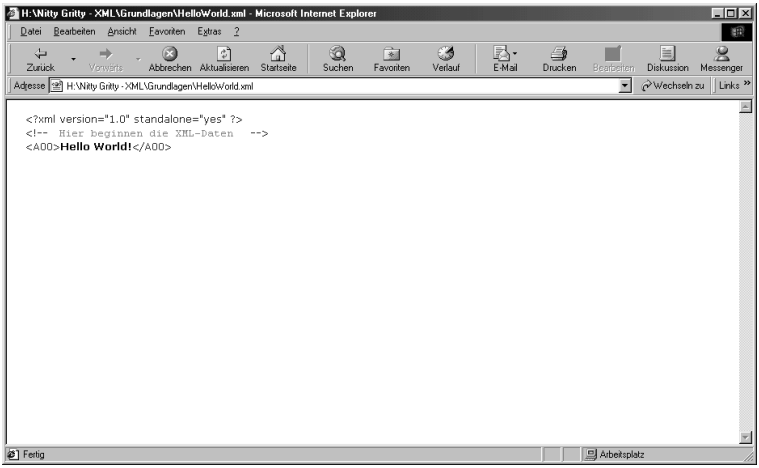

Bild 3.2: Ausgabe des XML-Dokuments

Sieht Ihr Ergebnis wie in Abbildung 3.2 aus, haben Sie alles richtig gemacht. Wie schon gesagt, weiß der Browser nicht, was er mit dem Element `A00` anfangen soll und das Ergebnis sieht nicht besonders ansprechend aus. Um das zu ändern, benutzen wir jetzt Stylesheets.

3.2 Stylesheets

Wie bereits erwähnt können Sie in XML alle benötigten Tags selbst definieren. Daher kann kein Browser im Vorhinein wissen, welche Tags Sie verwenden und welche Regeln für diese Tags gelten. Also brauchen wir einen Weg, um dem Browser mitzuteilen, wann er was mit dem Inhalt der Elemente machen soll. Im Augenblick benutzen wir dazu Cascading Stylesheets (CSS), die seit der ersten Generation der XML-fähigen Webbrowser unterstützt werden.

Das `Hello World!`-Beispiel enthält nur ein Tag, so dass wir nur dieses zu definieren brauchen. Dazu öffnen Sie wieder Ihren Editor, legen eine neue Datei mit dem Namen `helloworld.css` an und geben die folgende Zeile ein:

```
A00 {font-size: 48pt; font-weight: bold;}
```

Zum Speichern von Cascading Stylesheets wird die Dateiendung `.css` verwendet. Beim Abspeichern ist wieder darauf zu achten, dass auch diese Datei im Textformat abgespeichert wird.

Dieses Listing ist ein sehr einfaches Stylesheet, das festlegt, dass der Inhalt des Elements `A00` in Schriftgröße 48 und fett dargestellt werden soll.

Nachdem Sie das XML-Dokument und den CSS-Stylesheet erstellt haben, müssen Sie diese beiden Dateien noch miteinander verbinden, also den Browser anweisen, das Stylesheet auf das Dokument anzuwenden. Bitte ergänzen Sie Ihr XML-Dokument wie folgt:

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/css" href="helloworld.css"?>
<!-- Hier beginnen die XML-Daten -->
<A00>
Hello World!
</A00>
```

Die neue Anweisung `<?xml-stylesheet ?>` spezifiziert zum einen die verwendete Stylesheet-Sprache mit dem Attribut `type` und zum anderen mit dem Attribut `href` die URL, die angibt, wo das Stylesheet zu finden ist. In diesem Fall legt die Anweisung fest, dass ein Stylesheet mit dem Namen `helloworld.css` auf das Dokument angewendet werden soll, welches in der Stylesheet-Sprache CSS geschrieben worden ist.

Jetzt sollten Sie sich Ihr XML-Dokument noch einmal anschauen. Das Ergebnis ist besser als zuvor, oder?

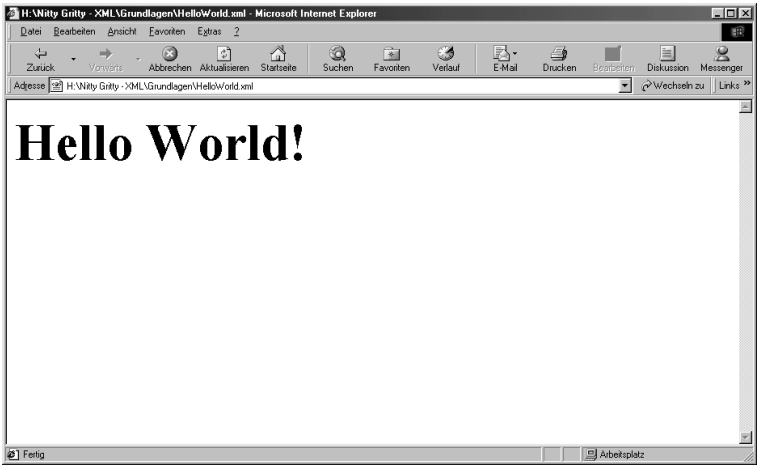

Bild 3.3: Ausgabe des XML-Dokuments mit Stylesheet

3.3 Personaldatei

Wir nehmen uns nun ein größeres Beispiel vor, um den Umgang mit den XML-Elementen zu erweitern und zu vertiefen. Zunächst werden wir die gegebenen Daten analysieren und in XML umwandeln.

Nehmen wir an, dass wir für die kleine Softwarefirma "ComputerSoftware GmbH" von Carl Computer eine Personaldatei erstellen wollen. Welche Daten haben wir? Die Angestellten liefern eine ganze Menge Daten: Namen, Vornamen, Geburtsdaten, Adressen, Telefonnummern, Handynummern usw. Aber auch die Daten bezüglich der Firma sind interessant, z.B. Abteilung und Position.

In der Tabelle haben wir einige fiktive Daten vorbereitet. Jede Ähnlichkeit mit realen Personen ist zufällig und nicht beabsichtigt. ☺

Name	Vorname	Straße	Wohnort	Geburtsdatum	Telefonnummer	Handy	Abteilung	Position
Ratlos	Ralf	Hauptstr. 134	44263 Dortmund	08.03.70	0231/51442	0199/2266555	Datenbankabteilung	Programmierer
Friedlich	Frida	Emil-Figge-Str. 1	58239 Schwerte	26.01.70	02304/785112	0199/2266553	Datenbankabteilung	Abteilungsleiterin
Hasenfuß	Hanna	Lüner Str. 12	44534 Lünen	12.05.67	02381/45331	0199/2266552	Geschäftsführung	Sekretärin
Computer	Carl	Ostkamp 10	44263 Dortmund	19.07.69	0231/911933	0199/2266551	Geschäftsführung	Geschäftsführer

Tabelle 3.1: Daten der Personaldatei

Name	Vorname	Straße	Wohnort	Geburtsdatum	Telefonnummer	Handy	Abteilung	Position
Tadel	Tina	Erlenhof 17a	44263 Dortmund	01.09.79	0231/987123	0199/2266558	Anwendungssoftware-abteilung	Programmierer
Mustermann	Max	Vinkestr. 32	59368 Werne	17.06.77	02389/4711	0199/2266557	Anwendungssoftware-abteilung	Programmierer
Sorglos	Siggi	Bahnhofstr. 98	44579 Bochum	30.10.74	0234/87521	0199/2266554	Anwendungssoftware-abteilung	Abteilungsleiter
Blödel	Bruno	Schlossallee 13	44263 Dortmund	23.12.75	0231/53557	0199/2266556	Datenbank-abteilung	Programmierer

Sie sehen, es kommen schnell ein paar Daten zusammen und jede Spalte definiert praktisch ein XML-Element. Wir brauchen also Elemente für Name, Vorname, Straße, Wohnort, Geburtsdatum, Telefonnummer, Handy, Abteilung und Position.

3.3.1 Umwandlung in XML

XML-Elemente können Text oder auch andere XML-Elemente enthalten. Obwohl das im Augenblick als schlechte Form gilt und möglichst vermieden werden sollte, können diese so genannten Child-Elemente (dt. Kind-`Elemente`) sowohl Text als auch Child-Elemente enthalten.

Um unsere Daten in XML umzuwandeln, müssen wir uns also zuerst überlegen, was worin enthalten ist, also welches Element Kind von einem anderen Element ist. Ist zum Beispiel ein(e) Angestellte(r) Kind in einer Abteilung oder hat eine Abteilung mehrere Kinder (Angestellte)? Die Antwort liegt nicht unbedingt auf der Hand, oder? Zudem kann ein(e) Angestellte(r) auch schon mal die eine Abteilung verlassen und in eine andere Abteilung wechseln. Es gibt so gesehen in XML meist mehr als einen Weg, seine Daten zu organisieren.

Wir haben uns dafür entschieden, dass einer Abteilung mehrere Angestellte angehören, da uns hier die Abteilungszugehörigkeit interessiert. Also fangen wir an.

3.3.2 Deklaration und Root-Element

Wie wir vorhin erwähnt haben, wird ein XML-Dokument an seiner Deklaration erkannt, die immer am Anfang einer XML-Datei steht.

```
<?xml version="1.0" standalone="yes"?>
```

Jedes wohlgeformte XML-Dokument muss ein Root-Element besitzen. Ein Root-Element zeichnet sich dadurch aus, dass es alle anderen Elemente einer XML-Datei umschließt. Das heißt, dass alle anderen Elemente Child-Elemente vom Root-Element sind. Als Root-Element werden wir Firma verwenden und alle weiteren Elemente werden zwischen dem Start-Tag `<Firma>` und dem End-Tag `</Firma>` Platz finden...

Bisher sieht unser Dokument so aus:

```
<?xml version="1.0" standalone="yes"?>
<Firma>
</Firma>
```

Da es sich um eine bestimmte Firma handelt, sollten wir den Namen der Firma noch in das XML-Dokument aufnehmen. Dazu legen wir das Element an, welches nachher den Firmennamen enthält. Natürlich gibt es mehrere Möglichkeiten bei der Definition des Elements und Sie haben dabei die freie Wahl, aber wenn Sie einfach nur `<Name>` zur Definition des Elements nehmen, kommen Sie schnell durcheinander. Allein in diesem kleinen Beispiel gibt es drei Möglichkeiten, ein Element mit `Name` zu definieren, also ist eine etwas speziellere Definition wie `<Name_Firma>` ratsam.

```
<?xml version="1.0" standalone="yes"?>
<Firma>
  <Name_Firma>
    ComputerSoftware GmbH
  </Name_Firma>
</Firma>
```

In den letzten Zeilen wurde mit Einrückungen für jedes einzelne Element gearbeitet, um Ihnen zu zeigen, dass das Element `<Name_Firma>` das Child-Element vom Root-Element `<Firma>` ist, und zum anderen, dass XML auch damit klarkommt. Außerdem möchten wir Ihnen einen guten Programmierstil nahelegen, mit dem Sie vor allem bei großen Dokumenten eine bessere Übersicht behalten.

```
<?xml version="1.0" standalone="yes"?><Firma><Name_Firma> Computer-
Software GmbH</Name_Firma></Firma>
```

Wenn Sie aber z.B. aus Platzgründen alle Elemente in einer Zeile zusammenfassen, hat XML auch damit keine Probleme. Es wird nur schwieriger, den Quelltext zu lesen.

```
<?xml version="1.0" standalone="yes"?>
<Firma>
  <Name_Firma> ComputerSoftware GmbH</Name_Firma>
</Firma>
```

Tatsächlich wird meistens eine Mischung aus beiden Extremen gewählt, solange die Lesbarkeit nicht darunter leidet, z.B. wenn ein Element komplett in eine Zeile passt.

3.3.3 Element Abteilung

Die Firma von Herrn Computer besteht aus drei Abteilungen, die wir wie folgt in unser XML-Dokument aufnehmen können.

```
<?xml version="1.0" standalone="yes"?>
<Firma>
<Name_Firma> ComputerSoftware GmbH</Name_Firma>
  <Abteilung>
    <Name_Abteilung> Geschaeftsfueührung</Name_Abteilung>
  </Abteilung>
  <Abteilung>
    <Name_Abteilung> Datenbankabteilung </Name_Abteilung>
  </Abteilung>
  <Abteilung>
    <Name_Abteilung>
      Anwendungssoftwareabteilung
    </Name_Abteilung>
  </Abteilung>
</Firma>
```

3.3.4 Element Person

Die restlichen Daten werden jetzt Child-Elemente vom Element `<Person>`, da wir keine weitere Verschachtelung mehr benötigen.

```
<Person>
  <Nachname> Computer</Nachname>
  <Vorname> Carl</Vorname>
  <Position> Geschaeftsfuehrer</Position>
  <Strasse> Ostkamp 10</Strasse>
  <Wohnort> 44263 Dortmund</Wohnort>
  <Geburtsdatum> 19.07.69</Geburtsdatum>
  <Telefonnummer> 0231/911933</Telefonnummer>
  <Handy> 0199/2266551</Handy>
</Person>
```

3.3.5 XML-Personaldatei im Ganzen

Bisher haben wir uns unsere Personaldatei nur in Teilen angesehen. Jetzt ist es an der Zeit, die Teile zusammenzufügen.

```
<?xml version="1.0" standalone="yes"?>
<Firma>
  <Name_Firma> ComputerSoftware GmbH</Name_Firma>
  <Abteilung>
    <Name_Abteilung> Geschaeftsfuehrung</Name_Abteilung>
    <Person>
      <Nachname> Computer</Nachname>
      <Vorname> Carl</Vorname>
      <Position> Geschaeftsfuehrer</Position>
      <Strasse> Ostkamp 10</Strasse>
      <Wohnort> 44263 Dortmund</Wohnort>
      <Geburtsdatum> 19.07.69</Geburtsdatum>
      <Telefonnummer> 0231/911933</Telefonnummer>
      <Handy> 0199/2266551</Handy>
    </Person>
    <Person>
      <Nachname> Hasenfuss</Nachname>
      <Vorname> Hanna</Vorname>
      <Position> Sekretaerin</Position>
      <Strasse> Luener Str. 12</Strasse>
      <Wohnort> 44534 Luenen</Wohnort>
      <Geburtsdatum> 12.05.67</Geburtsdatum>
      <Telefonnummer> 02381/45331</Telefonnummer>
      <Handy> 0199/2266552</Handy>
    </Person>
  </Abteilung>
  <Abteilung>
    <Name_Abteilung> Datenbankenabteilung</Name_Abteilung>
    <Person>
      <Nachname> Friedlich</Nachname>
      <Vorname> Frida</Vorname>
      <Position> Abteilungsleiterin</Position>
      <Strasse> Emil-Figge-Str. 1</Strasse>
      <Wohnort> 58239 Schwerte</Wohnort>
      <Geburtsdatum> 26.01.70</Geburtsdatum>
```

```
    <Telefonnummer> 02304/785112</Telefonnummer>
    <Handy> 0199/2266553</Handy>
  </Person>
  <Person>
    <Nachname> Ratlos </Nachname>
    <Vorname> Ralf </Vorname>
    <Position> Programmierer </Position>
    <Strasse> Hauptstr. 134</Strasse>
    <Wohnort> 44263 Dortmund</Wohnort>
    <Geburtsdatum> 08.03.70</Geburtsdatum>
    <Telefonnummer> 0231/51442</Telefonnummer>
    <Handy> 0199/2266555</Handy>
  </Person>
  <Person>
    <Nachname> Bloedel </Nachname>
    <Vorname> Bruno </Vorname>
    <Position> Programmierer </Position>
    <Strasse> Schlossallee 13</Strasse>
    <Wohnort> 44263 Dortmund </Wohnort>
    <Geburtsdatum> 23.12.75</Geburtsdatum>
    <Telefonnummer> 0231/53557</Telefonnummer>
    <Handy> 0199/2266556</Handy>
  </Person>
</Abteilung>
<Abteilung>
  <Name_Abteilung>
    Anwendungssoftwareabteilung
  </Name_Abteilung>
  <Person>
    <Nachname> Sorglos </Nachname>
    <Vorname> Siggi </Vorname>
    <Position> Abteilungsleiter </Position>
    <Strasse> Bahnhofstr. 98</Strasse>
    <Wohnort> 44579 Bochum </Wohnort>
    <Geburtsdatum> 30.10.74</Geburtsdatum>
    <Telefonnummer> 0234/87521</Telefonnummer>
    <Handy> 0199/2266554</Handy>
  </Person>
```

```
    <Person>
      <Nachname> Mustermann </Nachname>
      <Vorname> Max </Vorname>
      <Position> Programmierer </Position>
      <Strasse> Vinkestr. 32</Strasse>
      <Wohnort> 59368 Werne </Wohnort>
      <Geburtsdatum> 17.06.77</Geburtsdatum>
      <Telefonnummer> 02389/4711</Telefonnummer>
      <Handy> 0199/2266557</Handy>
    </Person>
    <Person>
      <Nachname> Tadel </Nachname>
      <Vorname> Tina </Vorname>
      <Position> Programmierer </Position>
      <Strasse> Erlenhof 17a </Strasse>
      <Wohnort> 44263 Dortmund </Wohnort>
      <Geburtsdatum> 01.09.79</Geburtsdatum>
      <Telefonnummer> 0231/987123</Telefonnummer>
      <Handy> 0199/2266558</Handy>
    </Person>
  </Abteilung>
</Firma>
```

Der Quelltext ist mittlerweile schon ziemlich umfangreich, oder? Die Daten in dieser Form darzustellen hat aber den Vorteil, dass sie selbstbeschreibend sind. Nebenbei sind wir auch dem zehnten Ziel des XML-Entwurfs treu geblieben: Die Knappheit von XML-Markup ist von minimaler Bedeutung.

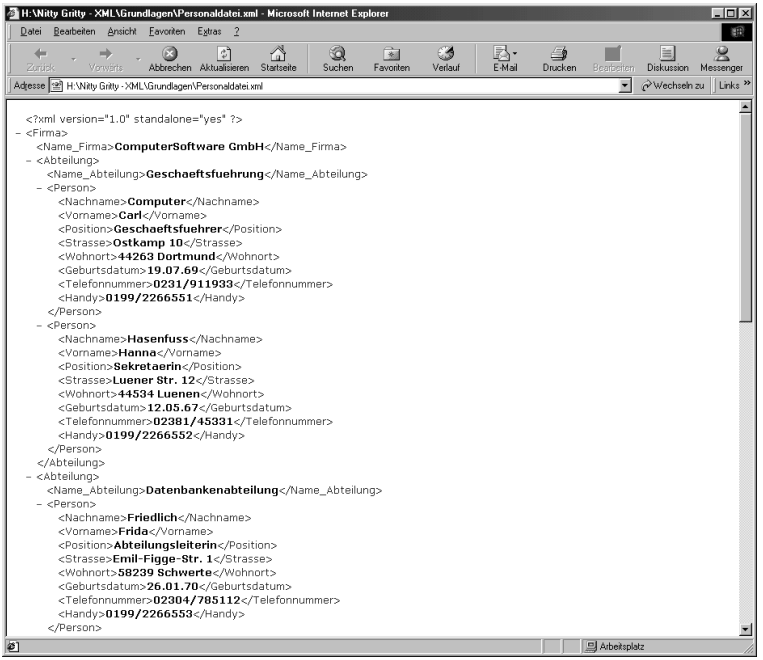

Bild 3.4: Ausgabe der Personaldatei

3.3.6 Personaldatei-Stylesheet

Wie in dem "Hello World!"-Beispiel erstellen wir nun ein Stylesheet für die Personaldatei. Nur haben wir nicht, wie im Beispiel zuvor, bloß <A00> als XML-Element, das wir definieren können, sondern wir können für alle dreizehn Elemente jeweils eine eigene Definition erstellen. Wir wollen uns aber auf drei beschränken, da ein CSS-Stylesheet für ein Element sich auf alle seine Child-Elemente weitervererbt. Nur eine eigene Stylesheet-Anweisung kann dies wieder aufheben.

Nehmen Sie nun wieder Ihren Editor und erstellen Sie ein Stylesheet aus den folgenden Zeilen. Als Name für dieses Stylesheet bietet sich hier Personaldatei.css an.

```
Name_Firma {font-size: 36pt; display: block; text-align: right; font-
weight: bold}
Name_Abteilung {font-size: 24pt; display: block; font-weight: bold}
Person {font-size: 12pt; display: block}
```

Die Attribute `font-size: 48pt` und `font-weight: bold` haben wir ja schon kennen gelernt, nur soll hier der Schriftgrad 36 benutzt werden. Mit `display: block` wird das Element als Block-Element dargestellt, d.h., dass sich das Element vom Inhalt absetzt und nicht mit dem Rest des Dokuments zusammenfließt. Das Element `Name_Firma` wird mit dem Attribut `text-align: right` rechtsbündig dargestellt. Nur Block-Elemente können mit `text-align:` rechtsbündig, linksbündig oder zentriert (`right, left, center`) ausgegeben werden.

Wenn Sie das Stylesheet erstellt haben, muss es nur noch mit dem XML-Dokument verbunden werden. Fügen Sie dazu noch die Anweisung `<?xml-stylesheet?>` zwischen XML-Deklaration und Root-Element ein:

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/css" href="personaldatei.css"?>
<Firma>
...
```

Bei dieser Anweisung wird angenommen, dass sich das Stylesheet in demselben Verzeichnis wie das XML-Dokument befindet. Komplette Pfadangaben sind in dieser Anweisung aber auch möglich. Ein Beispiel:

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/css" href="H:\Nitty Gritty - XML\Grundla-
gen\personaldatei.css"?>
<Firma>
...
```

Jetzt sollte Ihr XML-Dokument ungefähr so aussehen.

Bild 3.5: Ausgabe der Personaldatei mit Stylesheet

3.3.7 Attribute

Bisher haben wir alle Daten in einzelne Elemente gespeichert, aber das ist nicht die einzige Möglichkeit. XML-Elemente können wie HTML-Tags auch Attribute haben. Diese bestehen zum einen aus dem Namen, der nur einmal in jedem Element vorkommen kann, und aus dem Wert, der in Anführungsstrichen eingeschlossen ist, also ein Name-Wert-Paar (Name="Wert"). Bei der XML-Deklaration haben wir schon mit Attributen gearbeitet:

```
<?xml version="1.0" standalone="yes"?>
```

Sie verfügt über zwei Attribute, das Attribut `version` mit dem Wert `1.0` und das Attribut `standalone` mit dem Wert `yes`. Diese Attributsyntax ist auch auf die XML-Elemente in unserer Personaldatei anzuwenden. Statt wie bisher den Namen der Firma als Child-Element vom Element `Firma` laufen zu lassen

```
<Firma>
  <Name_Firma> ComputerSoftware GmbH</Name_Firma>
</Firma>
```

kann man den Name als Attribut ins Element `Firma` einfügen.

```
<Firma Name="ComputerSoftware GmbH">
</Firma>
```

So könnte man alle Child-Elemente als Attribut in das Root-Element aufnehmen, wenn die Attributnamen innerhalb eines Elements nicht eindeutig sein müssten. Deshalb sollten Sie beispielsweise das `Firma`-Element nicht so definieren:

```
<Firma Name="ComputerSoftware GmbH" Abteilung="Geschaeftsfuehrung" Abteilung="Datenbankabteilung" Abteilung="Anwendungssoftwareabteilung">
</Firma>
```


Ein anderer Grund gegen diese Aufteilung besteht darin, dass das Element `Abteilung` über weitere Child-Elemente verfügt und diese als Attribut nicht verarbeitet werden können. Durch die enge Bindung von einem Element und seinem Attribut ist es aber möglich, dass Sie `Name` als Namen des Attributs verwenden, ohne dass hier die Gefahr der Verwechslung der Namen, wie es bei den Elementen der Fall war, besteht. So kann man die Abteilungen etwa so codieren:

```
<Abteilung Name="Geschaeftsfuehrung">
</Abteilung>
<Abteilung Name="Datenbankabteilung">
</Abteilung>
<Abteilung Name="Anwendungssoftwareabteilung">
</Abteilung>
```

Das Element `Person` enthält eine große Anzahl von möglichen Attributen:

```
<Person Nachname="Computer" Vorname="Carl" Position="Geschaeftsfuehrer" Strasse="Ostkamp 10" Wohnort="44263 Dortmund" Geburtsdatum="19.07.69" Telefonnummer="0231/911933" Handy="0199/2266551">
</Person>
```

3.3.8 Leere Tags

Besteht ein Element nur aus Attributen und keinem Inhalt, benutzt man leere Tags als Abkürzung. Diese werden genauso behandelt wie nicht leere Tags. Sie unterscheiden sich von normalen Elementen dadurch, dass sie nur aus einem Tag, nämlich dem Start-Tag, bestehen und dieses Tag mit /> endet und nicht einfach nur mit >. So sieht zum Beispiel das Element `Person` so aus:

```
<Person Nachname="Computer" Vorname="Carl" Position="Geschaeftsfueh-
rer" Strasse="Ostkamp 10" Wohnort="44263 Dortmund" Geburtsda-
tum="19.07.69" Telefonnummer="0231/911933" Handy="0199/2266551"/>
```

3.3.9 XML-Personaldatei mit Attributen

Hier nun die gesamte Personaldatei im neuen Attributstil:

```
<?xml version="1.0" standalone="yes"?>
<Firma Name="ComputerSoftware GmbH">
  <Abteilung Name="Geschaeftsfuehrung">
    <Person Nachname="Computer" Vorname="Carl" Position="Geschaefts-
fuehrer" Strasse="Ostkamp 10" Wohnort="44263 Dortmund" Geburtsda-
tum="19.07.69" Telefonnummer="0231/911933" Handy="0199/2266551"/>
    <Person Nachname="Hasenfuss" Vorname="Hanna" Position="Sekretae-
rin" Strasse="Luener Str. 12" Wohnort="44534 Luenen" Geburtsda-
tum="12.05.67" Telefonnummer="02381/45331" Handy="0199/2266552"/>
  </Abteilung>
  <Abteilung Name="Datenbankenabteilung">
    <Person Nachname="Friedlich" Vorname="Frida" Position="Abteilungs-
leiterin" Strasse="Emil-Figge-Str. 1" Wohnort="58239 Schwerte"
Geburtsdatum="26.01.70" Telefonnummer="02304/785112" Handy="0199/
2266553"/>
    <Person Nachname="Ratlos" Vorname="Ralf" Position="Programmierer"
Strasse="Hauptstr. 134" Wohnort="44263 Dortmund" Geburtsda-
tum="08.03.70" Telefonnummer="0231/51442" Handy="0199/2266555"/>
    <Person Nachname="Bloedel" Vorname="Bruno" Position="Programmie-
rer" Strasse="Schlossallee 13" Wohnort="44263 Dortmund" Geburtsda-
tum="23.12.75" Telefonnummer="0231/53557" Handy="0199/2266556"/>
  </Abteilung>
  <Abteilung Name="Anwendungssoftwareabteilung">
    <Person Nachname="Sorglos" Vorname="Siggi" Position="Abteilungs-
leiter" Strasse="Bahnhofstr. 98" Wohnort="44579 Bochum" Geburtsda-
tum="30.10.74" Telefonnummer="0234/87521" Handy="0199/2266554"/>
    <Person Nachname="Mustermann" Vorname="Max" Position="Programmie-
rer" Strasse="Vinkestr. 32" Wohnort="59368 Werne" Geburtsda-
tum="17.06.77" Telefonnummer="02389/4711" Handy="0199/2266557"/>
    <Person Nachname="Tadel" Vorname="Tina" Position="Programmierer"
Strasse="Erlenhof 17a" Wohnort="44263 Dortmund" Geburtsda-
tum="01.09.79" Telefonnummer="0231/987123" Handy="0199/2266558"/>
  </Abteilung>
</Firma>
```

Natürlich stellen die beiden Möglichkeiten, alle Daten als Elemente oder alle Daten als Attribute darzustellen, absolute Extreme dar und es ist ein Mittelweg zwischen diesen Möglichkeiten wahrscheinlicher. Welche Methode Sie später wählen oder in welchem Umfang Sie diese Möglichkeiten mischen, hängt dann von Ihnen und den Anforderungen des Dokuments ab.

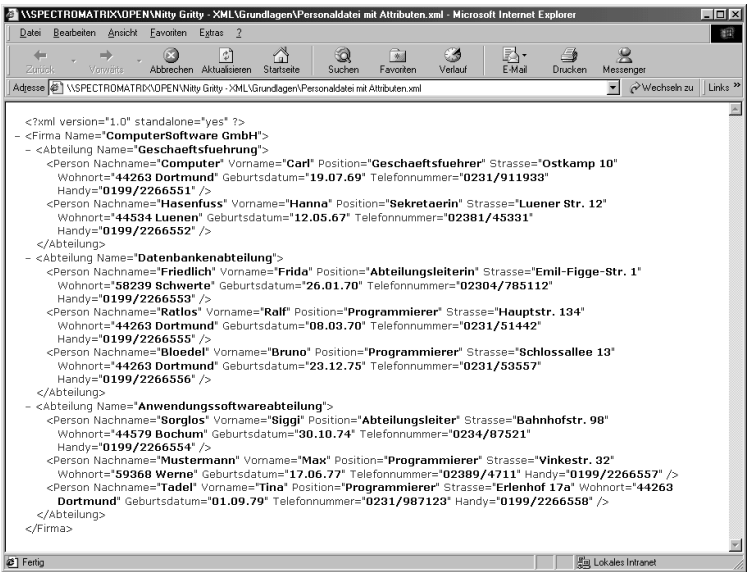

Bild 3.6: Daten der Personaldatei in Attributen

3.4 Wohlgeformtes XML

Anders als in HTML gibt es in XML so gut wie keine vordefinierten Tags, stattdessen haben Sie die Möglichkeit Ihre eigenen Tags bzw. Elemente nach Bedarf zu definieren. Damit die Elemente nicht willkürlich erstellt werden, gibt es in XML ein paar Regeln, die Sie befolgen müssen. Ein XML-Dokument, das diese Regeln befolgt, wird als wohlgeformt bezeichnet. Um von den XML-Prozessoren und XML-Browsern gelesen werden zu können, ist diese Wohlgeformtheit notwendig.

3.4.1 XML-Deklaration

Wenn eine XML-Deklaration vorhanden ist, muss sie im Dokument an erster Stelle stehen, weil der XML-Prozessor die ersten Bytes der Datei untersucht, um festzustellen, welcher Zeichensatz verwendet wird. Selbst Leerzeichen vor der XML-Deklaration könnten problematisch sein.

3.4.2 Root-Elemente

Ein wohlgeformtes XML-Dokument hat ein Root-Element, da XML nicht mehr als ein Element in der obersten Ebene erlaubt. Alle anderen Elemente müssen sich innerhalb des Start-Tags `<ROOT>` und dem End-Tag `</ROOT>` befinden. Natürlich können Sie Ihrem Root-Element auch einen anderen Namen geben.

3.4.3 Elemente

Elemente bestehen, wie Sie wissen, aus einem Start-Tag, seinem Inhalt und einem End-Tag. Ein Start-Tag beginnt mit `<`, gefolgt von seinem Namen, und endet mit `>`. Das End-Tag unterscheidet sich vom Start-Tag, da es mit `</` beginnt. Da XML davon ausgeht, dass eine öffnende spitze Klammer (`<`) immer am Anfang eines Tags und die schließende spitze Klammer (`>`) immer am Ende eines Tags steht, verwenden Sie diese bitte auch nur dort.

Während die Webbrowser bei HTML noch sehr tolerant waren und bei einem fehlenden End-Tag dennoch das Dokument anzeigten, wird man bei einem XML-Dokument nicht so viel Glück haben. Vergessen Sie ein abschließendes Tag, wird der Browser einfach einen Fehler als Ergebnis ausgeben.

Das Prinzip der Child-Elemente haben Sie ja schon kennen gelernt. Auch hier unterscheidet sich XML von HTML. In HTML ist es erlaubt, dass die Elemente bzw. Tags überlappen:

```
<center>
<h1>
Hello World!
</center>
</h1>
```

Ein XML-Browser ist hier nicht so nachsichtig. Dieser Code ist in XML nicht zulässig, da das schließende Tag `</center>` vor dem schließendem Tag `</h1>` steht – der Browser würde nur einen Fehler ausgeben. So sollte es richtig aussehen:

```
<center>
  <h1>
    Hello World
  </h1>
</center>
```

Natürlich kann man jederzeit ein leeres Tag ins Element einfügen:

```
  <h1>
    Hello <br/>World
  </h1>
```

Der Name des Elements kann nicht vollkommen willkürlich gewählt werden. Elementnamen müssen mit einem Buchstaben oder einem Unterstrich beginnen. Der Rest des Namens kann aus Buchstaben, Zahlen, Unterstrichen, Bindestrichen und Punkten bestehen. Ein Leerzeichen und Doppelpunkt darf im Namen nicht vorkommen. Ebenso sind alle Namen verboten, die mit `xml` oder `XML` anfangen, da diese Buchstabenfolgen für spätere XML-Bezeichnungen reserviert sind. Einige Beispiele für zulässige XML-Elementnamen:

```
<FIRMA>   </FIRMA>
<1.Abteilung>   <1.Abteilung>
<_root>   </_root>
<Carl_Computer>   </Carl_Computer>
<Name-Firma>   </Name-Firma>
<A00>   </A00>
```

Ein weiterer Unterschied zu HTML besteht darin, dass XML zwischen Groß- und Kleinschreibung unterscheidet. Sie dürfen zwar Groß- und Kleinschreibung verwenden, doch muss das Start-Tag mit dem End-Tag absolut identisch sein. So lässt sich zum Beispiel das Element `<A00>` nicht mit `</a00>` schließen.

3.4.4 Attribute

Auch bei den Attributen gibt es einige Regeln, an die Sie sich halten müssen. So verhält es sich mit den Attributnamen genauso wie bei den Namen der Elemente.

Ein Attributname darf nur mit einem Buchstaben oder einem Unterstrich beginnen, als folgende Zeichen im Namen sind Buchstaben, Zahlen, Unterstriche, Bindestriche und Punkte zugelassen. Wie bei den Elementnamen sind auch bei den Attributnamen keine Leerstellen und Doppelpunkte erlaubt und es sind alle Namen verboten, die mit den Buchstabenfolgen `xml` oder `XML` anfangen, da diese für spätere XML-Bezeichnungen reserviert sind.

Die Namen der Attribute dürfen pro Element nur einmal vorkommen. So ist zum Beispiel folgendes Element nicht zulässig:

```
<Abteilung Name="Datenbankabteilung" Name="Anwendungssoftwareabtei-
lung"/>
```

Da aber XML bei der Groß- und Kleinschreibung Unterschiede macht, wäre das folgende Element wiederum zulässig:

```
<Abteilung Name="Datenbankabteilung" name="Anwendungssoftwareabtei-
lung"/>
```

Diese Schreibweise ist aber nicht zu empfehlen, da man damit selbst schnell durcheinander kommen kann.

Bei den Attributwerten gibt es nicht so viele Einschränkungen. Attributwerte können Leerstellen enthalten und mit einer Zahl beginnen sowie fast alle Satzzeichen enthalten. Da XML-Attribute durch ein Anführungszeichen begrenzt werden, ersetzt man bei der Verwendung von Anführungszeichen innerhalb des Attributwerts das öffnende und das schließende Anführungszeichen jeweils durch einen Apostroph.

3.4.5 Kommentare

Der Inhalt von Kommentaren wird behandelt, als wäre er nicht vorhanden. Er dient zum Beispiel dazu, dass Sie sich Notizen machen oder noch nicht fertige Abschnitte des XML-Dokuments auskommentieren können. XML-Kommentare beginnen mit `<!--` und enden mit `-->`. Sie dürfen nicht in ein Tag eingefügt werden:

```
<?xml version="1.0" standalone="yes"?>
<A00 <!-- Hier beginnt das Element A00 -->>
  Hello World
</A00 <!-- Hier endet das Element A00 -->>
```

Zum Auskommentieren von Abschnitten wird das gesamte Element von dem Kommentar eingeschlossen und nicht nur ein Tag, da der XML-Browser dann entweder ein öffnendes oder schließendes Tag zu viel hätte; das gesamte Dokument wäre dann nicht mehr wohlgeformt und würde einen Fehler produzieren. Folgender Text wäre beispielsweise zulässig:

```
<?xml version="1.0" standalone="yes"?>
<ROOT>
  <A00>
    Hello World
  </A00>
<!--
  <B00>
    Hello XML
  </B00>
-->
</ROOT>
```

Da die Zeichenfolge (`--`) innerhalb des Kommentars nur beim Öffnen und Schließen des Kommentars auftreten darf, ist folgende Verschachtelung von Kommentaren nicht erlaubt:

```
<?xml version="1.0" standalone="yes"?>
<ROOT>
  <A00>
    Hello World
  </A00>
<!--
```

```
  <B00>
    <!-- Hello XML -->
  </B00>
-->
</ROOT>
```

3.4.6 Entity-Referenzen

Sie haben schon einige Zeichen kennen gelernt, die Sie in bestimmten Situationen nicht verwenden sollen, da XML diese Zeichen falsch interpretieren kann. So darf das kaufmännische Und (&) nur am Anfang von Entity-Referenzen stehen. Falls Sie aber dennoch eines dieser Zeichen verwenden müssen, stehen Ihnen die Entity-Referenzen zur Verfügung.

Entity-Referenzen sind Auszeichnungen, die durch diese Zeichen ersetzt werden, wenn das Dokument geparst wird. Wenn Sie zum Beispiel das Kleiner-als-Zeichen (<) verwenden wollen, welches von XML als Beginn eines Tags interpretiert werden würde, benutzen Sie einfach die Entity-Referenz `<`, jeder andere Ansatz wäre nicht mehr wohlgeformt.

Entity-Referenz	Zeichen
&amp;	&
&lt;	<
&gt;	>
&quot;	"
&apos;	'

Wenn Sie schon mit HTML gearbeitet haben, werden Sie die Entity-Referenzen schon kennen. Im Gegensatz zu den Entity-Referenzen in HTML müssen sie in XML mit einem Semikolon enden. Ein weiterer Unterschied zu HTML besteht darin, dass in XML nur diese fünf Entity-Referenzen vordefiniert sind.

3.4.7 CDATA

Stellen Sie sich vor, Sie müssten ein XML-Dokument schreiben, das eine Menge dieser Entity-Referenz-Zeichen enthält, zum Beispiel ein XML-Dokument über ein XML-Dokument:

```
&lt;?xml version="1.0" standalone="yes"?&gt;
&lt;A00&gt;
  Hello World
&lt;/A00&gt;
```


Sie müssten alle <-, >-, &-, "- und '-Zeichen austauschen. Ganz schön umständlich, oder? Und dies ist nur unser "Hello World!"-Beispiel. Bei einem C- oder Java-Quelltext wäre der Aufwand ähnlich. Abhilfe schafft hier ein `CDATA`-Abschnitt, der dazu verwendet wird, seinen Inhalt als reinen Text darzustellen. Somit werden die Zeichen <, >, &, " und ' auch als solche dargestellt. Ein CDATA-Abschnitt beginnt mit dem Tag `<![CDATA[` und endet mit `]]>`. Mit dem CDATA-Abschnitt kann man das obige Beispiel auch so darstellen:

```
<![CDATA[
<?xml version="1.0" standalone="yes"?>
<A00>
  Hello World
</A00>
]]>
```

Da der CDATA-Abschnitt mit dem schließenden Tag `]]>` endet, ist dies der einzige Text, der in einem solchem Abschnitt nicht vorkommen darf. Sollten Sie diesen Text einmal brauchen, werden Sie wohl für den gesamten Text-Abschnitt auf die Entity-Referenzen zurückgreifen müssen.

3.5 Zeichensätze

Ihnen ist bei unserem Beispiel mit der Personaldatei wahrscheinlich aufgefallen, dass wir die in Deutschland gebräuchlichen Sonderzeichen wie ä, ö, ü und ß nicht darstellen konnten. Das liegt daran, dass wir des XML-Dokument im ASCII-Format erstellt haben. ASCII ist das Akronym von `American Standard Code for Information Interchange` und beinhaltet die Zeichen, die zum Schreiben des amerika-

nischen Englisch nötig sind. Man verwendet zur Codierung der ASCII-Zeichen die Zahlen 0 bis 127. Wir benötigen aber zusätzlich die Zeichen von 128 aufwärts. Diese Voraussetzungen erfüllt der Zeichensatz Latin-1; er codiert zusätzlich den Bereich 128 bis 255 und ermöglicht die Darstellung von ä, ö, ü und ß. Aber sämtliche Zeichen, die es auf der Welt gibt, kann auch Latin-1 nicht darstellen.

Wenn man nicht auf ä, ö, ü und ß verzichten möchte, kann man dem XML-Prozessor mitteilen, dass wir im Latin-1-Zeichensatz arbeiten. Dazu erweitern Sie Ihre XML-Deklaration wie folgt:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
```

Sie können die Codierungsdeklaration auch in einer separaten Zeile direkt nach der XML-Deklaration einfügen:

```
<?xml encoding="ISO-8859-1"?>
```

Jetzt weiß der XML-Prozessor, dass Sie im Latin-1-Zeichensatz arbeiten. Das ist nicht immer ganz unproblematisch, da nicht alle XML-Prozessoren sämtliche Codierungen verarbeiten können. Sie können jetzt ä, ö, ü und ß einsetzen:

```
<?xml version="1.0" standalone="yes"?>
<?xml encoding="ISO-8859-1"?>
<Firma>
  <Name_Firma> ComputerSoftware GmbH</Name_Firma>
  <Abteilung>
    <Name_Abteilung> Geschäftsführung</Name_Abteilung>
    <Person>
      <Nachname> Computer</Nachname>
      <Vorname> Carl</Vorname>
      <Position> Geschäftsführer</Position>
      <Strasse> Ostkamp 10</Strasse>
      <Wohnort> 44263 Dortmund</Wohnort>
      <Geburtsdatum> 19.07.69</Geburtsdatum>
      <Telefonnummer> 0231/911933</Telefonnummer>
      <Handy> 0199/2266551</Handy>
    </Person>
...
```

und der Browser wird sie erkennen.

Name des Zeichensatzes	Sprache/Länder
US-ASCII	Englisch
UTF-8	Komprimierter Unicode
ISO-10646-UCS-2	Roher Unicode
ISO-8859-1	Latin-1, Westeuropa
ISO-8859-2	Latin-2, Osteuropa
ISO-8859-3	Latin-3, Südeuropa
ISO-8859-4	Latin-4, Nordeuropa
ISO-8859-5	ASCII plus Kyrillisch
ISO-8859-6	ASCII plus Arabisch
ISO-8859-7	ASCII plus Griechisch
ISO-8859-8	ASCII plus Hebräisch
ISO-8859-9	Latin-5, Türkisch
ISO-8859-10	Latin-6, ASCII plus nordische Sprachen
ISO-8859-11	ASCII plus Thai
ISO-8859-13	Latin-7, ASCII plus baltische Sprachen, insbesondere Lettisch
ISO-8859-14	Latin-8, ASCII plus Gälisch und Walisisch
ISO-8859-15	Latin-9, Latin-0, Westeuropa
ISO-2022-JP	Japanisch
ISO-2022-CN	Chinesisch
ISO-2022-KR	Koreanisch
Big5	Chinesisch, Taiwan
GB2312	Chinesisch, Hauptland China
KOI6-R	Russisch

Tabelle 3.2: Auszug aus den verbreiteten Zeichensätzen

Eine andere Möglichkeit wäre die Verwendung vom Unicode–Zeichensatz, der die Codierungsdeklaration nicht benötigt. Er codiert den Bereich von 0 bis 65535 und erfasst so einen weit größeren Bereich an Zeichen. XML an sich arbeitet mit Unicode-Zeichensatz bzw. mit UTF-8, einem komprimierten Unicode-Zeichensatz, der sofort die benötigten Zeichen kennt. Ihre Dokumente müssen in Unicode abgespeichert werden. Dokumente in Unicode sind ungefähr doppelt so groß wie solche die in ASCII oder Latin-1 geschrieben und abgespeichert worden sind, da Unicode zwei Byte für jedes einzelne Zeichen verwendet.

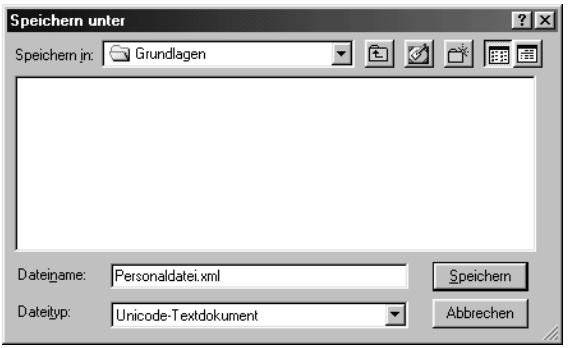

Bild 3.7: Speichern im Unicode-Format

Wenn Sie kein so großes Dokument haben wollen oder nicht in Unicode abspeichern können, können Sie das Dokument auch im ursprünglichen Zeichensatz belassen.

Jedes Unicode-Zeichen wird durch eine Zahl zwischen 0 und 65535 repräsentiert, die Sie mit Hilfe der Unicode-Zeichenreferenz einfügen können. Diese Zeichenreferenz besteht aus den beiden Zeichen `&#`, dem Zeichencode und einem abschließendem Semikolon. Beispielsweise wird ein `ä` folgendermaßen dargestellt: `&#228;`. Sie können den Zeichencode auch Hexadezimal (mit der Basis 16) darstellen, dem hexadezimalen Zeichencode wird dann nur ein `x` vorangestellt.

```
<?xml version="1.0" standalone="yes"?>
<?xml-stylesheet type="text/css" href="Personaldatei.css"?>
<Firma>
```

```
<Name_Firma> ComputerSoftware GmbH</Name_Firma>
<Abteilung>
  <Name_Abteilung>
    Gesch&#x00E4;ftsf&#x00FC;hrung
  </Name_Abteilung>
  <Person>
    <Nachname> Computer</Nachname>
    <Vorname> Carl</Vorname>
    <Position> Gesch&#x00E4;ftsf&#x00FC;hrer</Position>
    <Strasse> Ostkamp 10</Strasse>
    <Wohnort> 44263 Dortmund</Wohnort>
    <Geburtsdatum> 19.07.69</Geburtsdatum>
    <Telefonnummer> 0231/911933</Telefonnummer>
    <Handy> 0199/2266551</Handy>
  </Person>
...
```

Wie das obige Listing zeigt, müssen Sie die Unicode-Zeichenreferenz einfach nur einfügen.

Zeichen	Unicode-Zeichenreferenz als Dezimalzahl	Unicode-Zeichenreferenz als Hexadezimalzahl
Ä	&#196;	&#x00C4;
Ö	&#214;	&#x00D6;
Ü	&#220;	&#x00DC;
ß	&#223;	&#x;00DF
ä	&#228;	&#x00E4;
ö	&#246;	&#x00F6;
ü	&#252;	&#x00FC;

Tabelle 3.3: Auszug aus der Unicode-Zeichentabelle.

Welche Variante Sie zur Darstellung auch genommen haben, Ihr Ergebnis sollte bei jeder Variante wie Abbildung 3.8 aussehen.

Bild 3.8: Personaldatei

3.6 Namensräume in XML

Besonders bei Projekten mit mehreren Programmierern und bei öffentlichen XML-Dokumenten ist die Gefahr groß, dass Elementnamen doppelt vergeben werden und es so zu Problemen kommt. Um sicherzustellen, dass Elementnamen eindeutig sind und nicht zu Verwechslungen führen, also nicht doppelt definiert werden, wurden die Namensräume eingeführt. Mit den Namensräumen ist es jetzt ohne weiteres möglich, Teile von fremden Dokumenten zu verwenden.

Nehmen wir einmal an, dass sich in unserem Beispiel mit der Personalliste die Firma von Herrn Computer mit einer anderen Firma zusammenschließt und auch die andere Firma eine XML-Personalliste hat, die nun eingefügt wird. Auf Grund der vielen Möglichkeiten, ein XML-Dokument aufzubauen, kann es schnell passieren, dass doppelt vergebene Elementnamen kollidieren. Namensräume erweitern den Elementnamen und verhindern so eine Elementkollision.

Ein Namensraum lässt sich wie folgt bestimmen:

```
<Element_Name xmlns:Name="URL">
```

Die Deklaration des Namensraums wird in dem Root-Element vorgenommen. Also ist der Elementname der Name des Root-Elements. Es folgt der Name von dem Präfix `xmlns:` der die Deklaration des Namensraums einleitet. Diesem Präfix wird der Name des Namensraums zugewiesen, der durch die Zuweisung einer URL eindeutig wird. Für unser Personallistenbeispiel könnte die Nutzung eines Namensraums etwa so aussehen:

```
<?xml version="1.0" standalone="yes"?>
<CompSoft:Firma xmlns:CompSoft="http://www.ComputerSoftware.com">
  <CompSoft:Name_Firma>
    ComputerSoftware GmbH
  </CompSoft:Name_Firma>
  <CompSoft:Abteilung>
    <CompSoft:Name_Abteilung>
      Geschaeftsfuehrung
    </CompSoft:Name_Abteilung>
    <CompSoft:Person>
      <CompSoft:Nachname>Computer</ CompSoft:Nachname>
      <CompSoft:Vorname>Carl</CompSoft:Vorname>
      <CompSoft:Position>
        Geschaeftsfuehrer
      </CompSoft:Position>
      <CompSoft:Strasse>Ostkamp 10</CompSoft:Strasse>
      <CompSoft:Wohnort>44263 Dortmund</CompSoft:Wohnort>
      <CompSoft:Geburtsdatum>
        19.07.69
      </CompSoft:Geburtsdatum>
      <CompSoft:Telefonnummer>
        0231/911933
      </CompSoft:Telefonnummer>
      <CompSoft:Handy>0199/2266551</CompSoft:Handy>
    </CompSoft:Person>
...
```

Ihnen ist auch erlaubt, mehr als einen Namensraum zu verwenden. Zum Beispiel:

```
<CompSoft:Firma
 xmlns:CompSoft="http://www.ComputerSoftware.com"
xmlns:privat="http://www.carl-computer.de">
...
</CompSoft:Firma>
```