Teil 9
XML

56 XML – Eine Einleitung

Es ist nie zu spät, Vorurteile abzulegen.
– Henry David Thoreau, (1817–1862), US-amerikanischer Philosoph

In letzter Zeit ist XML immer öfter in aller Munde. Jedoch tauchen dabei auch gelegentlich Irrtümer und falsche Informationen auf. Was XML genau ist, erfahren Sie in diesem Kapitel.

56.1 Was ist XML?

Wie Sie bereits wissen, basiert HTML auf der Auszeichnungssprache SGML. Mittlerweile ist SGML etwas in die Jahre gekommen und kann als ausgereifte Sprache bezeichnet werden. Vor ein paar Jahren entschloss man sich dann, eine kleinere Version von SGML zu entwickeln, die nun als XML (Extensible Markup Language) bezeichnet wird. XML wird den Anforderungen an heutige Auszeichnungssprachen gerecht und hat sich inzwischen als *die* Auszeichnungssprache überhaupt etablieren können. XML ist also in keiner Weise der Nachfolger von HTML und basiert auch nicht auf XML.

Warum XML?

Wenn Sie einmal das Zusammenspiel von HTML und CSS etwas genauer betrachten, werden Sie feststellen, welche klare Trennung sich dadurch ergibt. HTML wird heutzutage eigentlich nur noch zur Strukturierung von Informationen genutzt. Dabei werden Überschriften und Textabsätze definiert, Tabellen erzeugt und an gewünschten Stellen auch Grafiken oder andere Objekte eingebunden. CSS dient dazu, diese Elemente so zu gestalten, dass sich ein einheitliches Bild ergibt. Diese Strukturierung der Daten, die durch HTML geschieht, macht sich XML zunutze, nur dass es dabei noch einen Schritt weiter geht.

So sollen mit XML nicht nur Strukturen wie HTML-Dokumente möglich werden, sondern – durch das Definieren von eigenen Elementen und Attributen – völlig neue Formate entstehen bzw. »ausgezeichnet« werden. Mit verschiedenen Techniken, die sich mit der Zeit gebildet haben, können mittlerweile sogar hoch flexible Datenbanksysteme mit XML aufgebaut werden, wenngleich diese Systeme noch relativ selten genutzt werden.

Dateiformate

Populärstes Beispiel für neue Dateiformate ist SVG, mit dem sich Vektorgrafiken ohne grafische Oberfläche von jedermann erzeugen lassen. Und selbst Microsoft hat die Bedeutung dieser Technologie entdeckt, wodurch mittlerweile auch Office-Dokumente wie Word und Excel mit XML definiert werden können.

56.2 Elemente und Attribute

Grundsätzlich kann ich an dieser Stelle ruhigen Gewissens behaupten, dass XML kein einziges Element oder Attribut vorschreibt oder gar kennt. Dies mag, mit dem Wissen um HTML im Hinterkopf, ein wenig irritierend erscheinen. Die Erklärung ist jedoch ganz einfach. XML legt lediglich fest, dass es Elemente geben kann, die außerdem Attribute erhalten dürfen. Dabei werden zwei Elementtypen unterschieden:

- Elemente, die ein Start- und ein Ende-Tag aufweisen und zwischen denen andere Elemente, Texte oder Daten notiert werden dürfen
- Elemente, die lediglich ein Startelement aufweisen und *keine* anderen Elemente, Texte oder Daten umschließen

Attribute werden immer im Start-Tag eines Elements notiert, wodurch sowohl Block- als auch Inline-Elemente mit Attributen versehen werden können

Spitzfindig

Alles, was XML in Hinblick auf Elemente vorschreibt, ist, dass sie in spitzen Klammern notiert werden müssen und dem Ende-Tag zusätzlich ein Schrägstrich vorangestellt wird, also:

```
<elementname>...</elementname>
```

Attribute werden im Start-Tag in folgendem Format geschrieben:

```
attributename="wert"
```

Ein Element kann beliebig viele Attribute erhalten.

```
<elementname attribut1="wert1" attribut2="wert2"
  attributN="wertN">...</elementname>
```

Handelt es sich um ein Element, das lediglich über ein Start-Tag verfügt, muss vor der schließenden spitzen Klammer ein Leerzeichen, gefolgt von einem Schrägstrich, eingefügt werden:

```
<elementname attribut="wert" />
```

Außerdem legt XML fest, dass es immer ein Wurzelelement geben muss, das alle anderen Elemente in dem Dokument vollständig umschließt.

Beispiel

Dazu ein kurzes Beispiel:

```
<?xml version="1.0"?>
<buchprojekte>
  <projekt name="Webseiten p.u.g">
    <titel>Webseiten programmieren und gestalten</titel>
    <auflage>2</auflage>
    <autor>Mark Lubkowitz</autor>
    <verlag bereich="Computing">Galileo Press</verlag>
```

```
    <cdrom beiliegend="ja" />
  </projekt>
</buchprojekte>
```

Listing 56.1 Beispiel für ein einfaches XML-Dokument

In dem Beispiel aus Listing 56.1 werden alle zuvor genannten Regeln erfüllt. Das Wurzelelement dieses Dokuments ist `buchprojekte`. Es umschließt alle in dem Dokument enthaltenen anderen Elemente.

Unterhalb des Wurzelelements wurde ein Element namens `projekt` notiert. Es verfügt sowohl über ein Start- und Ende-Tag als auch über ein Attribut namens `name`.

Dieses Element enthält weitere Elemente: `titel`, `auflage`, `verlag` und `cdrom`.

56.2.1 Processing Instructions

Eine Besonderheit im Vergleich zu HTML gibt es jedoch, und zwar die so genannten »Processing Instructions«. Im Listing 56.1 finden Sie eine solche Processing Instruction gleich in der ersten Zeile. Sie dient dazu, dem Programm, das das XML-Dokument interpretiert, bestimmte Anweisungen zu erteilen.

Processing Instructions werden dazu immer innerhalb der Zeichenfolgen `<?` und `?>` notiert. Direkt nach der Anfangszeichenfolge wird das Zielprogramm angegeben. In Listing 56.1 ist dies `xml`, was für das Programm steht, das das Dokument parst. Es teilt dem Programm mit, dass der Inhalt des folgenden Dokuments auf der XML-Version 1.0 basiert.

> Beachten Sie, dass in jedem XML- und auf XML basierenden Dokument immer die Processing Instruction
>
> `<?xml version="1.0"?>`
>
> notiert werden muss. Dies muss außerdem die erste Zeile in dem Dokument sein.

Eine Analogie lässt sich an dieser Stelle zu PHP ziehen. Alle Anweisungen, die vom PHP-Interpreter ausgeführt werden sollen, müssen zwischen den Zeichenfolgen `<?php` und `?>` aufgeführt werden. Als Zielprogramm wird dabei `php` angegeben, was für den PHP-Interpreter steht. PHP

56.3 Strukturierung, Bezeichnung und Kommentare

Besonders wichtig ist bei XML eine saubere Strukturierung. So muss z.B. unbedingt die Reihenfolge einhalten werden, in der die Start- und Ende-Tags notiert werden.

Den meisten Browsern ist es egal, ob in einem HTML-Dokument bei verschachtelten Elementen zuerst das Elternelement und erst danach das Kindelement geschlossen wird, z. B.

```
<b><i>Ein Text</b></i>
```

auch wenn dies grundsätzlich falsch ist. In einem XML-Dokument darf so etwas nicht passieren. Das Dokument muss folgendermaßen aussehen:

```
<b><i>Ein Text</i></b>
```

Aus diesem Grunde wird auch eine strenge Strukturierung der XML-Dokumente angewendet, um nicht aus Versehen diese Reihenfolge zu vertauschen.

Die wichtigste Regel: Vor jedem Start-Tag wird eine neue Zeile eingefügt und um zwei Spalten tiefer als das Elternelement eingerückt.

Folgen nach dem Start-Tag eines Elternelements weitere Elemente, dann wird das Ende-Tag ebenfalls in einer neuen Zeile, jedoch in der gleichen Spalte wie das Start-Tag notiert. Nur wenn das Element keine Kindelemente enthält, wird das Ende-Tag am Ende der Zeile eingefügt.

```
<element-1>
  <element-1.1>
    <element-1.1.1>Ein Text</element-1.1.1>
    <element-1.1.2>
      <element-1.1.2.1>Ein Text</element-1.1.2.1>
    </element-1.1.2>
  </element-1.1>
</element-1>
```

Hierarchie

Diese Struktur ermöglicht es erstens, Fehler zu vermeiden, und zwar bei der Reihenfolge, wie die Element geschlossen werden müssen, und erzeugt zweitens eine klare Übersicht darüber, in welcher Tiefe sich bestimmte Elemente befinden.

Diese Rangordnung, die durch das Verschachteln von Elementen entsteht, mag an dieser Stelle für Sie noch unwichtig sein. Möchten Sie später jedoch mit einer Programmiersprache darauf zugreifen, ist sie essenziell wichtig. Aus diesem Grunde habe ich die Elemente in dem vorangegangenen Beispiel auch mit »Kapitelnummern« bezeichnet, um Ihnen dies ein klein wenig zu verdeutlichen.

56.3.1 Bezeichnung

Bei der Bezeichnung der Elemente und Attribute müssen Sie selbstverständlich auch einige Regeln berücksichtigen. Hier beschränkten sich diese jedoch auf die zulässigen Zeichen und deren Platzierung.

- Sie dürfen alle Zeichen von A-Z und a-z im Bezeichner eines Elements oder Attributs verwenden. Auch der Unterstrich ist erlaubt. Eines dieser Zeichen muss außerdem an erster Stelle des Bezeichners notiert werden.
- Darüber hinaus sind die Zahlen von 0–9, der Bindestrich und der Punkt erlaubt, die jedoch erst ab der zweiten Stelle genutzt werden dürfen. Auch Umlaute und Akzente dürfen verwendet werden, wovon wegen der Internationalisierung jedoch abzuraten ist.
- Der Doppelpunkt sollte *nicht* verwendet werden.
- Die Länge der Bezeichner ist unbegrenzt. Sie sollten jedoch der Versuchung all zu langer Namen widerstehen.
- Die Zeichenfolge XML ist in keiner Schreibweise zu Beginn eines Bezeichners erlaubt.
- XML unterscheidet zwischen Groß- und Kleinschreibung. `Name` ist also nicht gleich `NAME` oder `nAME`. Dies ist übrigens die häufigste Fehlerquelle.

Korrekte Beispiele

Ein paar Beispiele für korrekte Bezeichner:

```
adresse
_adresse
ORT
ort2
plz-und-ort
_strasse_und_nr
```

Fehlerhafte Beispiele

Folgende Beispiele wären hingegen falsch:

```
.adresse
2orte
-hausnummer
xmlversion
```

Semantik

Versuchen Sie nach Möglichkeit, semantisch korrekte Bezeichner zu nutzen. Damit ist gemeint, dass die Bezeichner auch eine Beschreibung der Information darstellen sollen. Ein Beispiel:

```
<ort>Musterstr. 10, 55110 Musterstadt</ort>
```

Dies ist zwar syntaktisch korrekt und entspricht auch der »Wohlgeformtheit« eines XML-Dokuments, es ist semantisch jedoch nicht korrekt. Denn es werden neben dem Ort auch die Straße, Hausnummer und Postleitzahl gespeichert. Korrekt wären jedoch

```
<adresse>Musterstraße 10, 55110 Musterstadt</adresse>
```

oder;

```
<strasse>Musterstr.</strasse>
<hausnummer>10</hausnummer>
<plz>55110</plz>
<ort>Musterstadt</ort>
```

56.3.2 Kommentierung

Kommentare werden im Gegensatz zur Programmierung eher selten in XML-Dokumenten eingesetzt. Dies liegt nicht unbedingt daran, dass derjenige, der das Dokument erstellt hat, zu faul gewesen wäre. Im Gegenteil, je weniger Kommentare er eingesetzt hat, desto besser werden wahrscheinlich auch die Bezeichner für Elemente und Attribute gewählt sein. Ausnahmen bestätigen auch hier die Regel! Jedoch sind die Kommentare gelegentlich überaus hilfreich, z.B. dann, wenn Sie ein Dokument definiert haben, das nicht nur gleichmäßige Daten enthält, wie etwa eine auf XML basierende Datenbankstruktur für Kontaktdaten.

Je unterschiedlicher die Daten, desto mehr Kommentare

Als Beispiel wäre hier z.B. ein SVG-Dokument zu nennen. Da Sie viele verschiedene Formen wie Rechtecke und Kreise oder auch Farbverläufe und Pfade definieren können, ist es hier ratsam, Kommentare zu nutzen, um die Lesbarkeit eines solchen Dokuments für den menschlichen Betrachter zu erhöhen. Kommentare können jedoch auch zu Testzwecken genutzt werden, um bestimmte Daten temporär auszublenden.

Dabei gelten in XML die gleichen Regeln wie in HTML. Ein Kommentar wird mit

```
<!--
```

eingeleitet und mit

```
-->
```

abgeschlossen. Alles, was sich zwischen diesen beiden Zeichenfolgen befindet, wird vom interpretierenden Programm nicht berücksichtigt. Die beiden Zeichenfolgen müssen *nicht* in einer Zeile stehen, sondern können an beliebiger Stelle im Dokument eingesetzt werden.

Achten Sie jedoch möglichst auch bei Kommentaren darauf, korrekt einzurücken und die Zeichenfolge `<!--` wie ein Start-Tag und `-->` wie ein Ende-Tag zu behandeln.

56.4 DTD und Schema

Im Zusammenhang mit XML werden Sie des Öfteren auf die Begriffe DTD oder Schema treffen. DTD dürfte Ihnen noch von HTML bekannt vorkommen.

DTD

Die Document Type Definition (kurz DTD) legt bei einem XML-Dokument fest, welche Elemente verwendet werden, über welche Attribute sie verfügen, welche Art von Daten sie enthalten und an welcher Stelle sie eingesetzt werden dürfen.

Eine Weiterentwicklung der DTD sind die Schemata. Sie ermöglichen es nicht nur festzulegen, welche Art von Daten in einem Element notiert werden dürfen, sondern sogar, von welchem Typ sie sein müssen, z.B. Zeichenfolgen, Zahlen usw.

Schema

Die DTDs sind jedoch mittlerweile so sehr verbreitet, dass sie von den Schemata sicherlich nicht abgelöst werden. Darum hat sich eine gewisse Aufgabenteilung entwickelt. DTDs werden in der Regel dann genutzt, wenn es sich um ein Dokument handelt, das viele unregelmäßige Daten enthält wie z.B. SVG, WML[1] oder SMIL[2]. Schemata werden hingegen bei Dokumenten mit gleichmäßigen Daten genutzt, wie Dokumenten mit Adressdaten, Büchern oder RSS (dazu später mehr).

DTD oder Schema

56.5 Zusammenfassung

- XML ist keine Weiterentwicklung von HTML, sondern eine eigenständige Technologie
- XML unterscheidet bei Elementen und Attributen zwischen Groß- und Kleinschreibung
- DTD und Schemata stellen ein Regelwerk dar, wie ein auf XML basierendes Format einzusetzen ist und welche Elemente und Attribute erlaubt sind.
- Bei Elementen, die kein Ende-Tag aufweisen, müssen am Ende des Start-Tags vor der schließenden spitzen Klammer ein Leerzeichen sowie ein Schrägstrich notiert werden.

56.6 Fragen und Übungen

1. Was unterscheidet XML von HTML?
2. Bezeichnen in einem XML-Dokument `Text` und `text` das gleiche Element?
3. Welche Zeichen dürfen zum Benennen von Elementen und Attributen verwendet werden?
4. In welchen Fällen werden in der Regel DTDs und in welchen Schemata eingesetzt?
5. Was ist damit gemeint, wenn Elemente und Attribute »semantisch korrekt« sind?
6. Wie wird in XML-Dokumenten kommentiert?
7. Wofür werden Processing Instructions genutzt?
8. Was muss immer in einem XML-Dokument notiert werden?

1 WML = »Wireless Markup Language«. Wird für die Definition von Seiten für Geräte mit kleinen Bildschirmen genutzt, wie z.B. WAP.

2 SMIL = »Synchronized Multimedia Integration Language«. Ermöglicht die Einbindung und Steuerung von multimedialen Inhalten in Webseiten.

57 XHTML – Die nächste Generation von HTML

Der Politiker denkt an die nächsten Wahlen, der Staatsmann an die nächste Generation.
– William Gladstone, (1809–1898), engl. Pazifist und Politiker

Häufig trifft man auch auf die Bezeichnung XHTML, und das meistens im Zusammenhang mit XML. XHTML steht für »Extensible Hypertext Markup Language« – zu Deutsch »erweiterbare Hypertext–Auszeichnungssprache«. Die Verwandtschaft zwischen HTML und XHTML liegt aufgrund der Namensgebung sehr nahe. Und in der Tat: Die Ähnlichkeiten zwischen beiden Sprachen sind unverkennbar. Vereinfacht gesagt, ist XHTML eine Neudefinition von HTML.

Neue Sprachen

Sowohl mit SGML als auch mit XML ist es möglich, weitere Sprachstandards mit den so genannten DTDs zu definieren. Aufgrund der Beliebtheit von XML und der Forderung, HTML mit XML neu zu definieren, entstand im Januar 2000 die XHTML-1.0-Spezifikation. Trotz der großen Gemeinsamkeiten zwischen HTML und XHTML gibt es ein paar gravierende Unterschiede, die ich Ihnen im Folgenden noch genauer erklären werde.

57.1 Neu – und doch altbekannt

Tags

Atmen Sie erst einmal auf. Die meisten Tags, die Sie schon aus HTML kennen, sind erhalten geblieben. Neu ist jedoch die geringe Fehlertoleranz von XHTML. Die Start- und Ende-Tags sind nun case sensitive. C/C++-erfahrene Programmierer werden damit sicherlich eher etwas anfangen können. Case sensitive bedeutet, dass sehr genau auf die Groß- und Kleinschreibung der Tags geachtet wird. In XHTML **müssen** alle Tags und Attribute kleingeschrieben werden. Während Ihnen HTML noch die Wahl ließ, wie Sie ein Tag schreiben, bestraft XHTML die Großschreibung in der Regel mit einer Fehlermeldung.

Dies ist in HTML erlaubt:

```
<h1>...</h1>
<H1>...</H1>
<h1>...</H1>
<cItE>...</CiTe>
```

Dies ist in XHTML erlaubt:

```
<h1>...</h1>
<cite>...</cite>
```

In HTML ist also sowohl die Groß- und Kleinschreibung der Tags als auch eine Kombination erlaubt. In XHTML ist nur die Kleinschreibung erlaubt. Dies zwingt Sie dazu, standardkonformen Quelltext zu verfassen.

Elemente mit nur einem Tag

Aus HTML kennen Sie bereits Elemente, die lediglich ein Tag besitzen, nämlich das Start-Tag. Solche Elemente zeichnen keinen Text aus, sondern bewirken einen Effekt in der Ausgabe des Dokuments. So bindet das Element `img` eine Grafik ein, und `br` erzeugt einen Zeilenumbruch. In HTML reicht es, das Start-Tag zu notieren. In XHTML müssen jedoch aufgrund von XML alle Elemente abgeschlossen werden. Diejenigen Elemente, die kein Ende-Tag besitzen, werden aus diesem Grund mit einer Kurzschreibweise beendet:

In HTML:

```
<img src="...">
<hr color="red">
<br>
```

In XHTML lautet die Schreibweise:

```
<img src="..." />
<hr color="red" />
<br />
```

Elemente, die also nur ein Start-Tag besitzen, werden abgeschlossen, indem vor der schließenden spitzen Klammer ein Schrägstrich notiert wird. Achten Sie auch darauf, das Leerzeichen vor dem Schrägstrich zu notieren, da dieses ebenfalls erwartet wird.

Ebenenkonflikt

Noch verzeiht es HTML, wenn Sie z. B. zwei Elemente ineinander verschachteln, die Elemente jedoch in der falschen Reihenfolge wieder schließen:

```
<b>Dies ist ein <i>Text</b></i>
```

Dies ist ein Beispiel dafür. Das `i`-Element hätte vor dem `b`-Element geschlossen werden müssen. Die Browser zeigen die Formatierung in der Regel jedoch korrekt an. XHTML würde bei einem solchen Konstrukt das Handtuch werfen und jede weitere Arbeit verweigern. Sie müssen stattdessen also

```
<b>Dies ist ein <i>Text</i></b>
```

schreiben.

Werte in Anführungszeichen

Ich habe Ihnen verschwiegen, dass Sie in HTML Werte, die Sie an Attribute übergeben, nicht zwangsläufig in Anführungszeichen notieren müssen. Dies werden Sie auch auf vielen Webseiten wiederfinden. So wäre eigentlich auch die Angabe

```
<img src=grafik.png>
```

erlaubt. In XHTML hingegen müssen alle Werte in doppelten Anführungszeichen übergeben werden.

```
<img src="grafik.png" />
```

Fehlende Wertzuweisung

HTML kennt einige Attribute für Elemente, die keinen Wert zugewiesen bekommen müssen, etwa das Attribut `selected` des `option`-Elements. So dürfen Sie in HTML z.B. Folgendes schreiben:

```
<option name="n" value="w" selected></option>
```

In XHTML müssen Sie hingegen jedem Attribut einen Wert zuweisen. Also dürften Sie das `selected`-Attribut gar nicht notieren, es sei denn, Sie übergeben als Wert den Namen des Attributs. Dann wiederum akzeptiert XHTML die Angabe des Attributs:

```
<option name="n" value="w" selected="selected"></option>
```

Leerzeichen

Seien Sie vorsichtig mit der Verwendung von Leerzeichen bei Werten eines Attributs. Sollten Sie solche Leerzeichen wie im Attribut `name` verwenden, z.B.

```
<a name="eine marke">...</a>
```

kann dies, muss aber nicht, zu einem Fehler führen. Dies hängt vom Browser ab.

57.2 Erforderliche Angaben

Wie bei jedem so genannten »wohlgeformten« XML-Dokument müssen Sie auch in XHTML einige Angaben notieren, die zwingend erforderlich sind. Zu Beginn muss immer die XML-Processing-Instruction notiert werden. Diese lautet:

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

Sie zeigt an, dass die nachfolgenden Tags bzw. Elemente auf der Sprache XML basieren.

Da XML eine Auszeichnungssprache für andere Sprachen ist, muss dem Programm, das das Dokument öffnen und verarbeiten soll, noch mitgeteilt werden, welche Sprache verwendet wird und welche DTD eingesetzt werden soll.

Auch in XHTML gibt es die aus HTML bekannten Varianten `strict`, `transitional` und `frame`. Dementsprechend fallen auch die DTDs unterschiedlich aus:

Für die `strict`-Variante lautet die DTD:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "DTD/xhtml1-strict.dtd">
```

Für die `transitioal`-Variante lautet sie:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "DTD/xhtml1-transitional.dtd">
```

Und für die `frame`-Variante sieht sie so aus:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frame//EN"
  "DTD/xhtml1-frame.dtd">
```

Zusätzlich muss noch ein so genannter **Namensraum** definiert werden. Dieser wird im Start-Tag des Wurzelelements notiert (in diesem Fall `html`). Er lautet:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

Die Attribute `xml:lang` und `lang` haben hier nichts mit der Sprache zu tun, in der der Inhalt des Dokuments verfasst wurde, sondern mit der Sprache, mit der die Elemente bezeichnet wurden. Bei HTML und XHTML ist dies Englisch, daher die Abkürzung `en`.

Ein vollständiges XHTML-Dokument in der Variante `transitional` sieht also mindestens folgendermaßen aus:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title></title>
  </head>
  <body>
  </body>
</html>
```

Listing 57.1 Beispiel für ein vollständiges XHTML-Dokument

Die typische Dateiendung für XHTML-Dokumente lautet **.xhtml**. Diese Endung sollten auch Sie für Ihre Dokumente verwenden.

57.3 XHTML validieren

Aufgrund der sehr strengen Regeln von XHTML im Vergleich zu HTML kann es passieren, dass sich schneller Fehler einschleichen, als wenn Sie normales HTML verwenden. Dementsprechend wäre es sehr hilfreich – gerade wenn man einen einfachen Texteditor verwendet –, wenn man die Korrektheit des XHTML-Dokuments überprüfen könnte.

Das W3C bietet für solche Fälle einen Validator an, der Ihre Dokumente einem strengen Blick unterzieht und Ihnen jeden Fehler auflistet. Sie finden den Validator unter der Adresse

http://validator.w3.org/

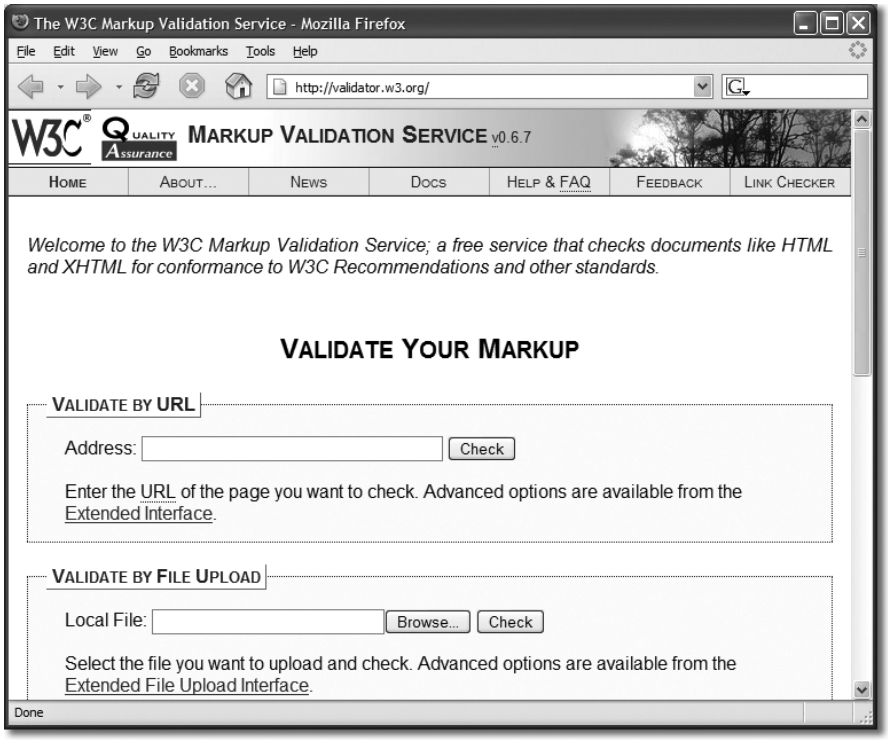

Abbildung 57.1 Der Markup Validation Service des W3C

Der Validator kann auf zwei verschiedene Arten genutzt werden:

- Entweder Sie geben in das Feld **Address** den URI Ihres zu validierenden Dokuments (egal ob HTML oder XHTML) ein, oder
- Sie wählen im Feld **Local File** ein Dokument auf Ihrem lokalen Rechner aus und laden es zum Validieren hoch.

Zusätzlich bietet das W3C einen Service an, der es Ihnen erlaubt, ein kleines Icon auf Ihrer Homepage einzubinden, das das Ergebnis der Validierung Ihrer Webseite anzeigt.

Abbildung 57.2 Viele Webseiten binden dieses Icon ein, um zu zeigen, dass sie HTML/XHTML-konform sind.

Um den Service nutzen zu können, müssen Sie lediglich den nachfolgenden Quelltext in Ihr Dokument einbinden:

```
<a href="http://validator.w3.org/check?uri=referer">
<img src="http://www.w3.org/Icons/valid-xhtml10"
 alt="Valid XHTML 1.0!" height="31" width="88"/>
</a>
```

57.4 Zusammenfassung

- XHTML ist HTML, das mit XML neu definiert wurde.
- Die Reihenfolge, in der Elemente geschlossen werden, muss unbedingt korrekt eingehalten werden. Das zuletzt geöffnete Element muss auch zuerst geschlossen werden.
- Das W3C bietet einen Service, um XHTML-Dokumente zu validieren.
- Existiert für ein Attribut kein Wert, muss der Bezeichner des Attributs als Wert notiert werden.

57.5 Fragen und Übungen

1. Was müssen Sie beim Notieren der Tags in Hinblick auf die Groß- und Kleinschreibung beachten?
2. Was müssen Sie beim Start-Tag beachten, falls das Element kein Ende-Tag besitzt?
3. Welche Angaben sind in einem XHTML-Dokument immer erforderlich?
4. Was müssen Sie bei Leerzeichen in Attributwerten beachten?

58 SVG – Scalable Vector Graphics

Wenn es nur eine einzige Wahrheit gäbe, könnte man nicht hundert Bilder über dasselbe Thema malen.
– Pablo Picasso, spanischer Maler, Grafiker und Bildhauer

SVG ist seit kurzer Zeit ein Schlagwort, auf das Sie an den verschiedensten Stellen stoßen werden. Was es mit SVG auf sich hat, werden Sie in diesem Unterkapitel erfahren.

58.1 Was sind Vektoren?

Ein Hauptproblem von Grafiken, die auf Webseiten verwendet werden, ist die fehlende Flexibilität. Das rührt daher, dass eine JPEG- oder PNG-Grafik aus Pixeln aufgebaut wird. Die Anzahl der Pixel hängt von der Größe der Grafik ab. Eine Grafik mit 400 Pixel Breite und 300 Pixel Höhe besitzt also 120.000 Pixel. Für jedes dieser Pixel ist innerhalb der Grafik eine Farbe definiert. Formen wie Linien, Rechtecke oder Kreise existieren also nur, weil bestimmte Pixel in einer bestimmten Anordnung die gleiche oder eine ähnliche Farbe aufweisen.

Vektoren sind anders

Bei Vektoren ist dies jedoch ganz anders. Eine auf Vektoren basierende Grafik besitzt keine Pixel, sondern Punkte in einem Koordinatensystem. Ein Vektor verbindet nun zwei Punkte in diesem Koordinatensystem miteinander. Diesem Vektor wird dann eine Farbe zugewiesen, und so entsteht dann eine Linie.

Optisch kaum Unterschiede

Auf den ersten Blick lassen sich Pixel-Grafiken kaum von Vektor-Grafiken unterscheiden. Das gilt aber auch nur, solange die beiden Grafiken in ihrer Originalgröße dargestellt werden. Vergrößern Sie eine Pixel-Grafik, wird diese sehr grobkörnig, da die einzelnen Pixel größer werden. Die Qualität der Grafik geht dann verloren. Wird hingegen eine Vektor-Grafik vergrößert, bleibt die Qualität gleich gut. Eine Vektor-Grafik bleibt also skalierbar, eine Pixel-Grafik nicht. Daher kommt auch die Bezeichnung »**S**calable **V**ector **G**raphics« (dt. *skalierbare Vektor-Grafiken*).

Beispiel

Sehen Sie sich dazu ein Beispiel an. Bei der Grafik aus Abbildung 58.1 lässt sich nicht erkennen, ob es sich um eine Pixel- oder eine Vektor-Grafik handelt, da die Grafik in ihrem Originalzustand dargestellt wird. Wie eine Vergrößerung der Grafik aussehen würde, wenn sie auf Vektoren basiert, können Sie in Abbildung 58.2 auf der linken Seite sehen. Die vergrößerte Darstellung, wenn es sich um eine Pixel-Grafik (gebräuchlichste Bezeichnung ist »Rastergrafik«) handelt, sehen Sie dann in Abbildung 58.2 auf der rechten Seite.

Abbildung 58.1 Die Grafik im Originalzustand. Es ist nicht erkennbar, ob sie pixel-oder vektorbasiert ist.

Abbildung 58.2 Vergrößerte Darstellung der Grafik aus Abbildung 58.1, links als Vektor-, rechts als Pixelgrafik

In Abbildung 58.2 ist der deutliche Qualitätsverlust bei Rastergrafiken zu erkennen.

Weiterer Vorteil

Ein weiterer Vorteil von vektorbasierten Grafiken ist die geringere Dateigröße. Die Grafik in Abbildung 58.1 beansprucht mit einer Größe von 300 × 300 als Vektor-Grafik lediglich ein paar Byte (ca. 0,5 kB), während sie als pixelbasierte Grafik bis zu 300 kB belegen kann. Vektor-Grafiken sind also auch in dieser Beziehung – wenn auch nur um eine Nasenlänge – im Vorteil. Erst ab einem gewissen Detailumfang einer Vektor-Grafik kann es passieren, dass sie größer als das Pixel-Pendant ist.

Verformbarkeit

Zu den bisher genannten Vorteilen der Vektor-Grafiken kommt noch einer hinzu: ihre Verformbarkeit. Während Pixel-Grafiken, wenn sie einmal erstellt worden sind, im Nachhinein nur sehr schwer bis so gut wie gar nicht veränderbar sind, können Vektor-Grafiken beliebig oft verändert werden, ohne dass mit Qualitätseinbußen gerechnet werden muss. So lässt sich z. B. die Farbe eines Textes in einer Vektor-Grafik mit zwei, drei Handgriffen verändern. Bei einer Pixel-Grafik ist dies wesentlich problematischer und bedarf oft einer zusätzlichen Nachbesserung.

Vektorgrafiken im Internet sind nichts Neues

Vektor-Grafiken werden im Internet schon seit geraumer Zeit in Form von Flash-Filmen (**.swf**) eingesetzt. Voraussetzung für die Bereitstellung von Flash-Filmen ist

jedoch in der Regel die Autorensoftware Macromedia Flash, die nicht unbedingt preiswert ist. Zudem muss der Nutzer zusätzlich ein Plug-in installiert haben, damit er solche Filme auch betrachten kann. Da SVG kein proprietärer Standard ist, sondern ähnlich wie HTML oder CSS durch das W3C entwickelt wurde und noch immer entwickelt wird, ist eine teure Autorensoftware nicht notwendig. Einzig und allein über Kenntnisse von XML und der darauf aufbauenden SVG sollte man verfügen. Das Plug-in ist jedoch noch immer erforderlich, da selbst Mozilla keine native Implementierung des SVG-Standards aufweist.

Um die Beispiele in diesem Abschnitt nachvollziehen zu können, sollten Sie ein Plug-in herunterladen und installieren. Das verbreitetste Plug-in ist der SVG Viewer von Adobe, den Sie unter **http://www.adobe.com/svg** kostenlos beziehen können.

58.2 Das erste SVG-Dokument

Das folgende Beispiel zeigt ein einfaches SVG-Dokument. Anschließend werde ich es Schritt für Schritt erklären. Kopieren Sie das SVG-Dokument aus Listing 3.1 in eine neue Datei, und speichern Sie die Datei unter einem Namen mit der Endung **.svg** ab.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <text x="10" y="60" font-family="Tahoma"
   font-size="50" fill="black">Das ist SVG!</text>
</svg>
```

Listing 58.1 Das erste SVG-Dokument

Öffnen Sie nun das Dokument mit einem Doppelklick. Wenn Sie versuchen, das Dokument über einen Webserver zu öffnen, wird dies scheitern. Den Grund dafür werden Sie in Abschnitt 58.3 erfahren.

Abbildung 58.3 Ausgabe des Listings 58.1 im Microsoft Internet Explorer mit dem Adobe SVG Viewer

Wenn die Ausgabe in Ihrem Browser derjenigen in Abbildung 58.3 entspricht, haben Sie soeben Ihr erstes SVG-Dokument erstellt. Sehen Sie sich den Quelltext einmal genauer an. Auf die beiden ersten Zeilen des Dokuments habe ich bereits in Kapitel 56 hingewiesen. Zuerst wurde die XML-Processing-Instruction notiert, die darauf hinweist, dass nun ein Dokument folgt, das auf XML basiert. Die in der zweiten Zeile folgende DTD gibt an, welche Auszeichnungssprache genau verwendet wird und beschreibt außerdem, welche Elemente und Attribute verwendet werden dürfen.

Die Elemente und Attribute

Dann folgt das Start-Tag des `svg`-Elements. `svg` ist das Wurzelelement und darf von keinem anderen Element umschlossen werden. Innerhalb des Start-Tags wird auch der XML-Namespace definiert. Damit wird erklärt, dass alle Elemente und Attribute, die innerhalb des Gültigkeitsbereichs des `svg`-Elements notiert werden, zu dem SVG-Namensraum gehören. Innerhalb des `svg`-Elements folgt dann ein `text`-Element. Letzteres erzeugt Textausgaben im Browser. `x` und `y` bestimmen die Position, an der der Text ausgegeben werden soll, `font-family` die Schriftart, `font-size` die Größe und `fill` die Farbe. Gewisse Ähnlichkeiten zu CSS bezüglich der Namensgebung der Attribute sind gewollt. Dazu später jedoch mehr.

Legen Sie sich jetzt am besten eine Vorlage für SVG-Dokumente an, die folgendermaßen aussehen sollte:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <!-- ... -->
</svg>
```

Listing 58.2 Vorlage für ein SVG-Dokument

Anstelle des Kommentars werden die SVG-Elemente notiert, die die Grafik dann definieren.

58.3 SVG in HTML

Wenn Sie versuchen, das SVG-Dokument aus Listing 58.1 über einen Webserver zu öffnen (also durch Angabe einer URI wie **http://localhost/list3.1.svg**), werden Sie feststellen, dass lediglich der Quelltext des Dokuments ausgegeben wird. Dies hängt mit dem MIME-Typ von SVG-Dokumenten zusammen. Einige Server senden nicht immer den korrekten MIME-Typ, so etwa der Apache Webserver. Anstatt dem Browser also mitzuteilen, dass nun ein Dokument vom Typ `image/svg+xml` folgt, geht der Server von einem ihm unbekannten Typ aus und schickt keinen MIME-Typ. Dies führt dazu, dass zwar die meisten Browser erkennen, dass es sich um ein XML-Dokument

handelt, sie können aber trotzdem ohne MIME-Typ damit nicht viel anfangen und geben entweder eine Fehlermeldung oder den Quelltext aus.

Rettung naht

Abhilfe schafft jedoch das HTML-Element `object`. Dieses Element ist in der Lage, jeden beliebigen Datentyp unter Angabe des MIME-Typs zu laden. Auch wenn der Webserver dann nicht den korrekten MIME-Typ sendet, wird das SVG-Dokument trotzdem dargestellt.

```
<html>
  <head>
    <title>Listing 3.2</title>
  </head>
  <body style="background-color:#336699;">
    <object data="list3.1.svg"
     type="image/svg+xml" width="100%"
     height="100%">
    </object>
  </body>
</html>
```

Listing 58.3 Einbettung eines SVG-Dokuments in ein HTML-Dokument mit Hilfe des object-Elements

Das `object`-Element erwartet dabei mehrere Attribute. Die beiden wichtigsten sind `data` und `type`. Mit `data` geben Sie die Quelle an, die dargestellt werden soll, und mit `type`, von welchem MIME-Typ diese Quelle ist. Zwar sind `width` und `height` optional, Sie sollten die beiden Attribute jedoch trotzdem mit angeben, da es keine Garantie gibt, dass der Browser die Breite und Höhe der darzustellenden Quelle auch korrekt erkennt.

Abbildung 58.4 Einbettung eines SVG-Dokuments mit Hilfe des object-Elements

Der blaue Rand und das eingebettete SVG-Dokument stammen übrigens von dem HTML-Dokument selbst, da dessen Hintergrundfarbe auf `#336699` festgelegt wurde.

58.4 Grundformen

Während Sie bei der Verwendung der GD-Library spezielle Funktionen aufrufen mussten, um Formen zu zeichnen, wird dies in SVG alles über Elemente und Attribute geregelt. Für Linien, Rechtecke und Kreise stehen verschiedene Elemente bereit, die sich durch die vielen Attribute aber deutlich besser gestalten lassen.

58.4.1 Text ausgeben

Text können Sie in SVG mit dem Element `text` ausgeben. Der auszugebende Text wird dabei innerhalb des Gültigkeitsbereichs des Elements notiert (also genauso wie in HTML). Die Koordinaten, an denen der Text ausgegeben werden soll, werden durch die Attribute `x` und `y` definiert. Der Text wird dann mit der linken unteren Ecke an diesen Koordinaten platziert.

Beachten Sie, dass dies von der Positionierung von Text mit der GD-Library abweicht. Dort wird die linke obere Ecke des Textes an der angegebenen Position ausgerichtet.

Schriftart Die Schriftart des Textes definieren Sie mit dem Attribut `font-family`. Während Sie in CSS einfach nur einen Schrifttyp (`serif`, `sans-serif`, `monospace` etc.) angeben konnten, ist dies in SVG nicht möglich. Hier müssen Sie eine bestimmte Schriftart angeben. Sie sollten sich deshalb vorerst auf weit verbreitete Schriftarten beschränken, wie etwa Arial, Times oder Courier.

Schriftgröße Auch bei dem Attribut `font-size` sind Sie in Ihrem Handlungsspielraum stark eingeschränkt, da Sie die Größe der Schrift lediglich in Punkt angeben können. Da Ihnen nur diese Möglichkeit bleibt, müssen Sie jedoch keine Angaben zur Maßeinheit machen.

Textfarbe Die Farbe können Sie mit `fill` angeben. Dabei können Sie sowohl ein Farbwort wie **`black`**, `yellow`, `lime` oder Ähnliches, aber auch Hex-Tripel-Werte wie `#FF0000` oder `#FFFF00` angeben. Es mag verwirrend erscheinen, dass die Textfarbe nicht über `color` festgelegt wird. Stellen Sie sich den auszugebenden Text einfach als eine gezeichnete Fläche vor, die anschließend mit einer Farbe gefüllt wird. Darüber hinaus wird auch in verktorbasierten Grafikprogrammen wie Adobe Illustrator oder Macromedia Flash die Textfarbe mit der Füllfarbe festgelegt.

```
<text x="100" y="100" font-family="Arial"
 font-size="40" fill="#996633">Text (Arial,40,#996633)</text>
```

Dieses Beispiel würde an den Koordinaten (100,100) einen Text in der Schriftart Arial mit der Schriftgröße 40 und in der Farbe `#996633` ausgeben.

style-Attribut Alternativ können Sie die Angaben zur Schriftformatierung auch mit dem `style`-Attribut vornehmen. Dies ist auch der Grund, warum die Eigenschaften zur Formatierung

eine starke Ähnlichkeit zu CSS aufweisen. Es erleichtert das Lernen von SVG und vereinfacht die Erstellung von solchen Grafiken, da Sie nicht zusätzlich spezielle Bezeichner verwenden müssen. Das obige Beispiel würde dann folgendermaßen geschrieben werden:

```
<text x="100" y="100" style="font-family:Arial;
 font-size:40; fill:#996633">Text (Arial,40,#996633)</text>
```

Außenlinie

Einen speziellen Effekt können Sie auch mit dem Attribut `stroke` erzeugen. Wenn Sie die Farbe von `fill` auf die Hintergrundfarbe setzen und die Farbe für `stroke` auf eine andere, erhalten Sie einen Text, der nur eine Außenlinie besitzt. Mit `stroke-width` können Sie dann noch die Dicke der Umrandung definieren.

```
<text x="100" y="100" style="font-family:Arial;
 font-size:40; fill:#FFFFFF; stroke:#000000;
 stroke-width:3">Text</text>
```

Schriftdicke und -stil

Auch die Attribute `font-weight` und `font-style` sind in SVG bekannt. So können Sie mit `font-weight:bold;` die Schriftdicke auf fett setzen und mit `font-style:italic;` den Schriftstil auf kursiv.

Sehen Sie sich dazu ein vollständiges Beispiel und dessen Ausgabe im Browser an.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <text x="30" y="60" style="font-family:'Times New Roman';
   font-size:40; fill:#999999; font-style:italic">filled/italic
  </text>
  <text x="30" y="100" style="font-family:Arial;
   font-size:40; fill:#FFFFFF; stroke:#000000">
   outlined</text>
  <text x="30" y="140" style="font-family:'Courier New';
   font-size:40; fill:#FFFF00; stroke:#000000;
   font-weight:bold">stroked/bold</text>
</svg>
```

Listing 58.4 Verschiedene Texte mit unterschiedlichen Formatierungen

Abbildung 58.5 Ausgabe des Listing 58.4 im Internet Explorer mit dem SVG Viewer

58.4.2 Linien

Linien können Sie in SVG mit dem Element `line` erzeugen. Dabei müssen Sie den Start- und den Endpunkt festlegen. Die Attribute `x1` und `y1` definieren den Startpunkt und `x2` und `y2` den Endpunkt. Auch hier stehen die Attribute `stroke` oder `stroke-width` zur Verfügung, um die optische Gestaltung der Linie zu beeinflussen. Mit `stroke` beeinflussen Sie die Farbe der Linie und mit `stroke-width` die Dicke. Das Element `line` besitzt übrigens kein Ende-Tag.

```
<line x1="10" y1="20" x2="200" y2="50" stroke="#000000" />
  <line x1="10" y1="40" x2="200" y2="70" stroke="#999999"
        stroke-width="5" />
```

Das erste `line`-Element würde eine Linie in der Farbe Schwarz vom Punkt (10,20) zum Punkt (200,50) zeichnen. Das zweite würde eine Linie vom Punkt (10,40) zum Punkt (200,70) zeichnen, diesmal in der Farbe `#999999` und mit einer Dicke von 5.

Mehrfachlinie

Alternativ können Sie auch das Element `polyline` verwenden, mit dem Sie mehr als nur zwei Punkte miteinander verbinden können. Die einzelnen Punkte werden dann in der Reihenfolge `x1 y1 x2 y2 ... xn yn` an das Attribut `points` übergeben. Normalerweise werden dabei jedoch der Start- und der Endpunkt ebenfalls miteinander verbunden, und die daraus resultierende Fläche wird mit einer Farbe gefüllt. Wenn Sie dies nicht möchten, sollten Sie dem Attribut `fill` den Wert `none` zuweisen.

```
<polyline points="60 10 70 100 80 10" fill="none" stroke="black"
          stroke-width="5" />
```

Dieses Beispiel würde zwei Linien zeichnen. Eine vom Punkt (60,10) zum Punkt (70,100) und eine von (70,100) zum Punkt (80,10) – beide in der Farbe Schwarz und mit der Dicke 5.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <line x1="10" y1="20" x2="200" y2="50"
   stroke="#000000" />
  <line x1="10" y1="40" x2="200" y2="70"
   stroke="#999999" stroke-width="5" />
  <polyline points="60 10 70 100 80 10 90 100 100 10"
   fill="none" stroke="black" stroke-width="5" />
</svg>
```

Listing 58.5 Zeichnen verschiedener Linien mit SVG

Abbildung 58.6 Ausgabe des Listing 58.5 im Internet Explorer mit dem SVG Viewer

58.4.3 Rechtecke

Das Element `rect` definiert in einer SVG-Grafik ein Rechteck. Dabei sind jedoch gewisse Mindestangaben erforderlich. Zum einen ist dies die Position, an der die linke obere Ecke des Rechtecks platziert werden soll, und zum anderen die Breite und die Höhe des Rechtecks. Natürlich sollte auch eine Farbe angegeben werden. Die Position wird mit den Attributen `x` und `y` angegeben, die Breite und Höhe mit `width` und `height`. Die Angabe einer Farbe kann entweder mit `fill` oder `stroke` bzw. mit beiden Attributen erfolgen.

```
<rect x="100" y="100" width="400" height="300"
      fill="red" />
```

Dieses Beispiel erzeugt, beginnend an der Position (100,100), ein Rechteck, das 400 Punkte breit und 300 Punkte hoch ist. Die Flächenfarbe des Rechtecks ist Rot.

Runde Ecken

Die Ecken eines solchen Rechtecks lassen sich auch abrunden. Dafür werden die Attribute `rx` und `ry` benötigt. Dabei beschreibt `ry` die Höhe der Abrundung und `rx` die Breite. Damit die Rundung also zu sehen ist, müssen beide Attribute mindestens einen Wert aufweisen, der größer als 0 ist. Für eine symmetrische Rundung muss sowohl `rx` als auch `ry` der gleiche Wert zugewiesen werden.

```
<rect x="30" y="30" rx="10" ry="20" width="100" height="100"
      fill="blue" />
```

Das Beispiel erzeugt an der Position (30,30) ein 100*100 großes, blaues Rechteck mit asymmetrisch abgerundeten Ecken.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="50" y="50" width="150" height="100"
   fill="red" />
  <rect x="225" y="50" rx="20" ry="20" width="100"
    height="100" fill="none" stroke="black"
    stroke-width="5"/>
```

```
  <rect x="50" y="175" rx="35" ry="15" width="275"
    height="50" style="fill:yellow; stroke:black;
    stroke-width:2"/>
</svg>
```

Listing 58.6 Ausgabe verschiedenartiger Rechtecke

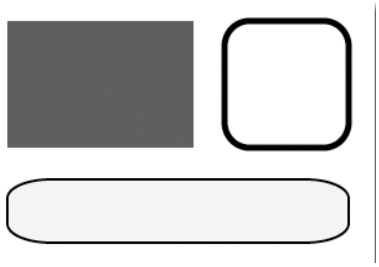

Abbildung 58.7 Ausgabe des Listing 58.6 im Internet Explorer mit dem SVG Viewer

58.4.4 Kreise und Ellipsen

Kreise werden mit dem Element `circle` definiert. Um einen Kreis zu zeichnen, müssen Sie den Mittelpunkt des Kreises und den Radius angeben. Der Mittelpunkt wird anhand der Attribute `cx` und `cy` angegeben und der Radius mit `r`. Natürlich sind auch hier wieder `fill`, `stroke` und `stroke-width` erlaubt.

```
<circle cx="100" cy="100" r="80" fill="green" />
```

Dieses Beispiel erzeugt an der Position (100,100) einen Kreis mit einem Radius von 80 in der Farbe Grün.

Ellipse zeichnen

Für eine Ellipse müssen Sie das Element `ellipse` und anstelle des Attributs `r` die Attribute `rx` und `ry` angeben. Diese definieren den horizontalen (`rx`) und den vertikalen Radius (`ry`).

```
<ellipse cx="150" cy="100" rx="125" ry="75" fill="none"
         stroke="black" stroke-width="3" />
```

Zeichnet eine Ellipse mit einem schwarzen Außenrahmen. Der Mittelpunkt der Ellipse liegt an der Position (150,100). Die beiden Radien betragen horizontal 125 und vertikal 75.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/
2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <circle cx="50" cy="50" r="30" fill="yellow"
   stroke="black" />
  <ellipse cx="110" cy="120" rx="100" ry="25"
```

```
   style="fill:none; stroke:black; stroke-width:5" />
</svg>
```

Listing 58.7 Ausgabe eines Kreises und einer Ellipse

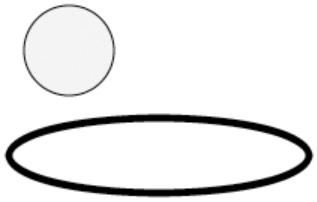

Abbildung 58.8 Ausgabe des Listing 58.7 im Internet Explorer mit dem SVG Viewer

58.5 Farben

Neben den bisher vorgestellten Formatierungsmöglichkeiten wie `fill`, `stroke` oder `stroke-width` gibt es noch weitere, die ich an dieser Stelle kurz erläutern werde.

58.5.1 Transparenz

Natürlich gibt es auch in SVG die Möglichkeit, Transparenz zu definieren. Dies gilt sowohl für die Füllfarbe als auch für die Farbe der Außenlinie. Die Transparenz der Füllfarbe wird mit `fill-opacity` und die Transparenz der Außenlinie mit `stroke-opacity` definiert. Für beide Attribute gilt ein Wertebereich von 0.0 (transparent) bis 1.0 (deckend).

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/
2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="40" y="40" width="100" height="100" fill="black" />
  <circle cx="140" cy="90" r="50" fill="yellow"
   fillopacity="0.8" />
</svg>
```

Listing 58.8 Transparenz von Füllfarben

Abbildung 58.9 Ausgabe des Listing 58.8 im Internet Explorer mit dem SVG Viewer

Da die Farbe für Texte ebenfalls mit einer Füllfarbe festgelegt wird, kann auch für Text der Grad der Transparenz mit `fill-opacity` festgelegt werden. Testen Sie doch einmal das folgende Beispiel:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <rect x="40" y="40" width="100" height="100" fill="black" />
  <circle cx="140" cy="90" r="50" fill="yellow"
   fillopacity="0.8" />
  <text x="100" y="100" style="font-family:Arial; font-
    size:40; fill:#FF0000; stroke:#000000; stroke-width:3;
    fill-opacity:0.7">Tranparenter Text</text>
</svg>
```

Listing 58.9 Transparente Formen und Texte

58.5.2 Lineare Farbverläufe

Mit dem Element `linearGradient` können Sie einen linearen Farbverlauf erzeugen. Mit dem Attribut `id` müssen Sie dem Farbverlauf eine eindeutige ID geben. Diese ID wird später benötigt, um den Farbverlauf für ein Element als Füllfarbe referenzieren zu können. Mit dem Element `stop` werden dann die Farben für den Verlauf definiert. Der Startpunkt als Prozentwert wird mit `offset` angegeben und die Farbe mit `stop-color`.

```
<linearGradient id="farbverlauf1">
  <stop offset="0%" stop-color="#FF6600" />
  <stop offset="100%" stop-color="#FFFF66" />
</linearGradient>
```

Dieses Beispiel definiert einen Farbverlauf mit der ID `farbverlauf1`. Der Farbverlauf beginnt bei 0 % mit der Farbe `#FF6600` und endet bei 100 % mit der Farbe `#FFFF66`.

Füllfarbe mit der URL angeben

Bei einer Form können Sie diesen Farbverlauf dann als Füllfarbe verwenden. Dabei wird anstelle eines Farbnamens `url` und in Klammern dahinter die ID des Farbverlaufs mit vorangestelltem Rautezeichen notiert:

```
<rect fill="url(#farbverlauf1)" x="10" y="10" width="100"
  height="50" />
```

Je mehr `stop`-Elemente Sie notieren, desto mehr Verläufe gibt es dann innerhalb des Farbverlaufs.

```
<linearGradient id="farbverlauf2">
  <stop offset="0%" stop-color="#99CCFF" />
```

```
  <stop offset="25%" stop-color="#003366" />
  <stop offset="100%" stop-color="#99CCFF" />
</linearGradient>
```

Dies erzeugt ebenfalls einen Farbverlauf. Bei 0% besitzt er die Farbe `#99CCFF`, bei 25% `#003366` und bei 100% des Verlaufs wieder `#99CCFF`.

Das folgende Listing verwendet beide Farbverläufe als Füllfarbe für je ein Rechteck.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <linearGradient id="farbverlauf1">
    <stop offset="5%" stop-color="#FF6600" />
    <stop offset="95%" stop-color="#FFFF66" />
  </linearGradient>
  <linearGradient id="farbverlauf2">
    <stop offset="0%" stop-color="#99CCFF" />
    <stop offset="25%" stop-color="#003366" />
    <stop offset="100%" stop-color="#99CCFF" />
  </linearGradient>
  <rect x="40" y="40" width="200" height="50"
    fill="url(#farbverlauf1)" stroke="black"
    stroke-width="2" />
  <rect x="40" y="100" width="200" height="50"
    fill="url(#farbverlauf2)" stroke="black"
    stroke-width="2" />
</svg>
```

Listing 58.10 Füllung zweier Rechtecke mit einem Farbverlauf

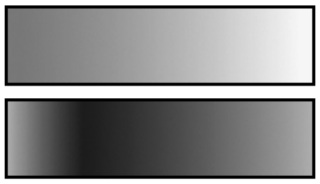

Abbildung 58.10 Ausgabe des Listing 58.10 im Internet Explorer mit dem SVG Viewer

58.5.3 Radiale Farbverläufe

Es ist auch möglich, einen radialen Farbverlauf zu definieren, und zwar mit `radialGradient`. Die restliche Definition unterscheidet sich nicht von der Definition eines linearen Farbverlaufs. Mit dem Attribut `id` wird dem Farbverlauf eine ID vergeben, und mit `stop`-Elementen werden dann die Farben definiert.

```
<radialGradient id="radFarb1">
  <stop offset="0%" stop-color="FF6600" />
  <stop offset="100%" stop-color="FFFF66" />
</radialGradient>
```

Ein vollständiges Beispiel:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/
2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <radialGradient id="radFarb1">
    <stop offset="0%" stop-color="#FFFF66" />
    <stop offset="100%" stop-color="#FF6600" />
  </radialGradient>
  <radialGradient id="radFarb2">
    <stop offset="0%" stop-color="#99CCFF" />
    <stop offset="25%" stop-color="#003366" />
    <stop offset="100%" stop-color="#99CCFF" />
  </radialGradient>
  <circle cx="100" cy="100" r="75" fill="url(#radFarb1)"
    stroke="black" stroke-width="2" />
  <rect x="200" y="25" width="150" height="150"
    fill="url(#radFarb2)" stroke="black" stroke-width="2" />
</svg>
```

Listing 58.11 Zwei radiale Farbverläufe, die einem Kreis und einem Rechteck zugewiesen werden

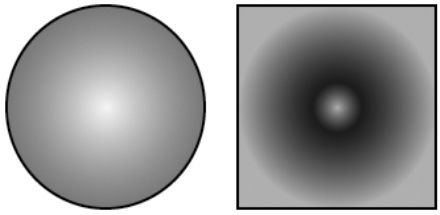

Abbildung 58.11 Ausgabe des Listing 58.11 im Internet Explorer mit dem SVG Viewer

Auch bei radialen und linearen Farbverläufen ist die Definition der Transparenz möglich. Diese Definition muss jedoch bei einem der Elemente vorgenommen werden, das den Farbverlauf verwendet. Das zu verwendende Attribut ist dann natürlich `fill-opacity`.

58.6 Pfade

Zusätzlich zu den einfachen Grundformen bietet SVG jedoch auch die Möglichkeit, Pfade zu definieren.

58.6.1 Einfache Pfade

Pfade sind sehr viel flexibler als Grundformen, da sie viel komplexere Figuren erzeugen können und sich nicht nur auf Linien, Rechtecke oder Kreise beschränken. Sie können mit einem Pfad sogar solche Grundformen zeichnen, auch wenn dies manchmal in einem Zahlenwirrwarr endet. Das Element zum Zeichnen eines Pfades ist `path`. Gezeichnet wird ein Pfad, indem mit bestimmten Steuerzeichen und Koordinaten die Verbindungen zwischen den einzelnen Punktpaaren definiert werden. Diese Steuerzeichen und Koordinaten werden an das Attribut `d` übergeben.

```
<path d="M 100 100 L 200 200" stroke="black" stroke-width="4" />
```

Erklärung

Dieses `path`-Beispiel zeichnet vom Punkt (100,100) zum Punkt (200,200) eine Linie mit einer Dicke von 4 und in der Farbe Schwarz. In diesem Beispiel finden Sie zwei Steuerzeichen: `M` und `L`. `M` ist die Abkürzung für `moveto` und definiert sozusagen den Startpunkt des Pfads ausgehend von der linken oberen Ecke der Zeichenfläche (also (0,0). `L` ist die Abkürzung für `lineto` und zeichnet eine gerade Linie zum angegebenen Punkt (ebenfalls in Relation zum Punkt (0,0)). Versuchen Sie sich den Pfad einfach als Stift vorzustellen, der am Punkt (100,100) angesetzt wird und dann eine Gerade zum Punkt (200,200) zeichnet.

Neben den beiden Steuerzeichen `M` und `L` gibt es natürlich noch weitere:

- `z`
  Steht für »closepath« und bedeutet, dass der Startpunkt und der letzte angegebene Punkt miteinander verbunden werden
- `H`, `h`
  Steht für »horizontal« und zeichnet eine horizontale Linie zum nächsten Punkt. `H` weist darauf hin, dass der folgende Punkt eine absolute Angabe ist, und `h`, dass der Punkt relativ zum vorherigen Punkt angegeben wird.
- `V`, `v`
  Steht für »vertical« und zeichnet eine vertikale Linie zum nächsten Punkt. `V` gibt einen absoluten Punkt an und `v` einen zum vorherigen Punkt relativen Punkt.
- `Q`, `q`
  Steht für »quadratic bézier curveto« und zeichnet anstelle einer geraden Linie eine Kurve vom letzten zum folgenden Punkt. `Q` gibt wieder einen absoluten und `q` einen relativen Punkt an.

Sehen Sie sich dazu ein paar Beispiele an.

```
<path d="M 200 100 L 300 300 L 100 300 z" fill="yellow"
 stroke="black" stroke-width="4" />
```

Dreieck zeichnen

Zeichnet ein Dreieck, indem bei dem Punkt (200,100) angesetzt und eine erste Linie zum Punkt (300,300) gezeichnet wird. Dann folgt eine zweite Linie zum Punkt (100,300). Durch die Angabe von `z` werden der letzte Punkt (100,300) und der Startpunkt (200,100) miteinander verbunden, der Pfad wird also geschlossen.

```
<path d="M 100 100 H 300 V 300 H 100 z" fill="yellow" stroke="black"
  stroke-width="4" />
```

Rechteck zeichnen

Dies Beispiel zeichnet ein Rechteck unter Verwendung der Steuerzeichen `H` und `V`. Der Vorteil dieser beiden Steuerzeichen ist, dass Sie, wenn Sie eine horizontale oder vertikale Linie zeichnen wollen, nur die Hälfte der Koordinate angeben müssen. Die Linie wird dann in Bezug auf den vorherigen Punkt gezeichnet. Anstatt also x und y anzugeben, müssen Sie bei einer horizontalen Linie nur x und bei einer vertikalen nur y angeben. Der Rest des Punkts wird vom vorherigen Punkt abgeleitet.

58.6.2 Kurven

Das Zeichnen von Kurven ist ein klein wenig komplizierter. Die Anweisung

```
<path d="M 100 100 Q 200 100 200 200" fill="none" stroke="black"
  stroke-width="4" />
```

Erklärung

zeichnet eine Kurve vom Punkt (100,100) zum Punkt (200,200). Der Punkt (200,100), den Sie im Wert des Attributs `d` finden, ist der so genannte Kontrollpunkt. Wenn die Kurve gezeichnet wird, orientiert sich die Kurve an diesem Kontrollpunkt. Würde dieser Punkt fehlen, würde lediglich eine Gerade gezeichnet. Im Folgenden sehen Sie noch ein detaillierteres Beispiel zu den Kurven:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/
2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <line x1="100" y1="100" x2="200" y2="100" stroke="gray"
   stroke-width="1" />
  <line x1="200" y1="100" x2="200" y2="200" stroke="gray"
   stroke-width="1" />
  <line x1="200" y1="200" x2="200" y2="300" stroke="gray"
   stroke-width="1" />
  <line x1="200" y1="300" x2="300" y2="300" stroke="gray"
   stroke-width="1" />
  <circle cx="200" cy="100" r="5" fill="none"
   stroke="red" stroke-width="1" />
  <circle cx="200" cy="300" r="5" fill="none"
   stroke="red" stroke-width="1" />
  <path d="M 100 100 Q 200 100 200 200 200 300 300 300"
```

```
   fill="none" stroke="black" stroke-width="4" />
</svg>
```

Listing 58.12 Zeichnen einer Kurve mit Darstellung der Kontrollpunkte

In Abbildung 58.12 können Sie die Ausgabe des Listing 58.12 sehen. Die dicke schwarze Linie ist die gezeichnete Kurve. Die dünnen grauen Linien stellen die Funktion der Kontrollpunkte dar. Die Kontrollpunkte selbst werden durch die kleinen Kreise dargestellt.

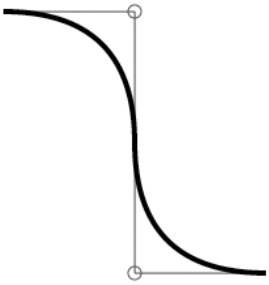

Abbildung 58.12 Ausgabe des Listing 58.12 im Internet Explorer mit dem SVG Viewer

Beim Zeichnen einer solchen Kurve ist die Reihenfolge der Punkte äußerst wichtig. Nach dem Steuerzeichen `Q` ist jedes erste Punktpaar ein Kontrollpunkt und jedes zweite Punktpaar ein Punkt der Linie, zu dem die Kurve gezeichnet werden soll. Bei

```
d="M 100 100 Q 200 100 200 200 200 300 300 300"
```

sind also die Punkte (200,100) und (200,300) Kontrollpunkte und (200,200) und (300,300) Punkte der Kurve.

Dies wirkt anfangs sehr kompliziert und bedarf auch einiger Übung. Je öfter Sie aber solche Kurven zeichnen, desto einfacher wird später auch die Handhabung.

58.6.3 Kreisbögen

Zum Zeichnen von Kreisbögen benötigen Sie das Steuerzeichen »`A`«, das für »elliptical arc« steht. Auch hier wird wieder zwischen Groß- und Kleinschreibung unterschieden. So gibt `A` an, dass mit absoluten Angaben gearbeitet wird, und `a` bedeutet, dass relative Angaben verwendet werden. Jedoch ist das Zeichnen von Kreis- bzw. Ellipsenbögen in SVG nicht so leicht wie in PHP oder Perl mit der GD-Library.

Radien und Flags

Nach dem Steuerzeichen erwartet SVG eine bestimmte Anzahl von Werten. Zuerst erwartet es den x- und den y-Radius der Ellipse. Soll der Bogen von einem Kreis stammen, müssen Sie ebenfalls beide Radien angeben, dann aber mit jeweils gleichen Werten. Anschließend folgt die Rotation des Kreises bzw. der Ellipse auf der x-Achse.

Erlaubt sind sowohl positive als auch negative Werte. Mit zwei verschiedenen Flags wird die Ausgabe des Bogens beeinflusst. Erlaubt sind jeweils die beiden Werte 0 oder 1. Dazu etwas später mehr. Zum Schluss folgt der Punkt, bis zu dem der Kreisbogen gezeichnet werden soll. Der Startpunkt resultiert aus dem vorherigen Punkt.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <path d="M 100 100 A 100 100 0 0,1 200 200" fill="none"
    stroke="black" stroke-width="3" />
</svg>
```

Listing 58.13 Ausgabe eines Kreisbogens in SVG

In der Abbildung 58.13 können Sie den Kreisbogen sehen, der durch das Listing 58.13 erzeugt wird. Zur Verdeutlichung wurden die einzelnen Punkte, zwischen denen der Kreisbogen gezeichnet wird, als graue Kreise dargestellt und die Radien mit einer grauen Linie eingezeichnet.

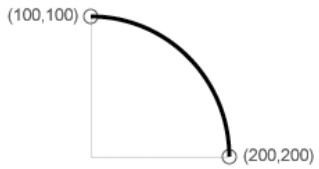

Abbildung 58.13 Ausgabe des Listings 58.11 im Browser (Die grau gezeichneten Elemente wurden nachträglich zur Verdeutlichung hinzugefügt.)

Der Kreisbogen wird durch folgende Anweisung im Attribut `d` des `path`-Elements gezeichnet:

```
A 100 100 0 0,1 200 200
```

Erklärung Nach dem Steuerzeichen `A` folgen die beiden Radien des Kreises, dessen Werte jeweils 100 betragen. Da der Kreis nicht rotiert werden soll, wurde 0 angegeben. Dann folgen die beiden Flags, die auf `0,1` gesetzt wurden. Das Komma ist optional, d.h., Sie können es eintragen, müssen es aber nicht tun. Der Lesbarkeit halber sollten Sie es aber trotzdem notieren. Das erste Flag ist das *large-arc-flag* und das zweite das *sweep-flag*. Die beiden letzten Zahlen geben den Endpunkt des Kreisbogens an. Der Startpunkt wurde bereits durch `M 100 100` definiert.

Sinn der Flags Die Flags haben bei dem Zeichnen von Kreisbögen eine ganz besondere Rolle. Stellen Sie sich einfach zwei Kreise vor. Beide Kreise schneiden sich an zwei Punkten: dem Startpunkt des Kreisbogens und dem Endpunkt des Kreisbogens.

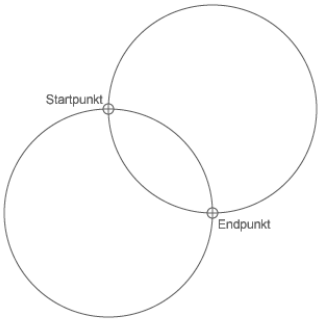

Abbildung 58.14 Die Schnittpunkte der beiden Kreise bilden den Start- und den Endpunkt des Kreisbogens.

Mit den beiden Flags wird nun definiert, welcher der Kreisbögen, die aus den Überschneidungen resultieren, gezeichnet wird. Wird das *large-arc-flag* auf 1 gesetzt, wird der größere der beiden Kreisbögen gezeichnet, und bei 0 der kleinere. Mit dem Setzen des *sweep-flags* auf 1 oder 0 wählen Sie den Kreis aus, dessen Kreisbogen gezeichnet wird. In Abbildung 58.15 finden Sie alle vier Möglichkeiten schematisch dargestellt.

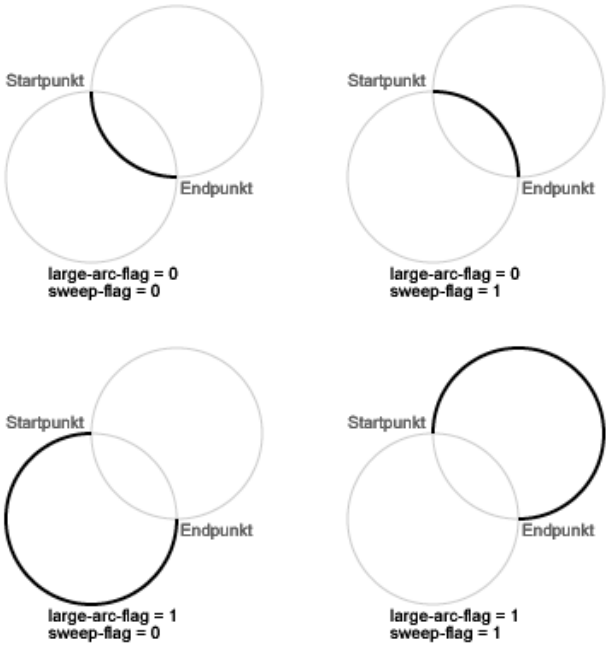

Abbildung 58.15 Darstellung der vier Möglichkeiten, welcher Kreisbogen gezeichnet wird

Abbildung 58.14 und Abbildung 58.15 wurden mit SVG erzeugt. Sie finden die beiden Quelltexte auf der dem Buch beiliegenden CD-ROM im Verzeichnis **x:\misc**.

Werkzeug

Das *sweep-flag* ist jedoch ein wenig trickreich. Es orientiert sich immer in Zeichenrichtung des Bogens. Die Zeichenrichtung selbst bestimmt sich durch den Start- und den Endpunkt. Es wird also immer vom Start- zum Endpunkt gezeichnet. Stellen Sie sich nun einfach vor, Sie würden auf dem Startpunkt stehen und in Richtung Endpunkt gucken. Setzen Sie dann das *sweep-flag* auf 0, wird der vom Startpunkt aus gesehen rechte Bogen gezeichnet, Bei 1 wird der vom Startpunkt aus gesehen linke Bogen gezeichnet.

Kreissektoren

Bis jetzt wurde immer ein Kreisbogen gezeichnet. Was ist aber, wenn Sie einen Kreissektor zeichnen möchten? Dies ermöglichen Ihnen das Steuerzeichen `L`, mit dem Sie eine Linie zeichnen können, und das Steuerzeichen `z`, das den Startpunkt und den letzten Punkt mit einer geraden Linie verbindet.

Um nun z.B. ein Viertelsegment eines Kreises zu zeichnen, müssen Sie nach der Definition des Kreisbogens lediglich eine Linie zum Mittelpunkt des Kreises definieren und das Ganze mit `z` abschließen.

```
<path d="M 200 100 A 100 100 0 0,1 300 200 L 200 200 z" fill="yellow"
  stroke="black" stroke-width="2" />
```

Erklärung

Dieses Beispiel bewegt den Zeichenstift zuerst zum Punkt (200,100). Dann folgt die Definition des Kreisbogens. Die beiden Radien des Kreises erhalten als Wert jeweils 100. Da der Kreis nicht auf der x-Achse rotiert werden soll, wird 0 angegeben. Das *large-arc-flag* wird auf 0 gesetzt, so dass der kleinere der beiden Kreisbögen gezeichnet wird, und das *sweep-flag* erhält 1, da vom Startpunkt aus gesehen der linke Kreisbogen gezeichnet werden soll. Der Endpunkt des Kreisbogens ist der Punkt (300,200). Dann wird eine Linie zum Mittelpunkt des Kreises gezogen. Der Mittelpunkt weist die Koordinaten (200,200) auf, und der Pfad wird nun mit `z` abgeschlossen. Abbildung 58.16 zeigt die Ausgabe dieses Pfades im Browser.

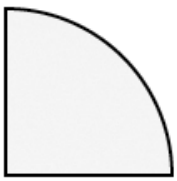

Abbildung 58.16 Ein Kreissegment, das mit Hilfe eines Pfades und eines Kreisbogens gezeichnet wurde

58.7 SVG mit PHP

Ähnlich wie die Ausgabe von HTML mit PHP und der `echo`-Anweisung ist es auch möglich, SVG-Dokumente zu erzeugen und auszugeben. Dies ist natürlich auch mit Perl zu realisieren, und die folgenden Beispiele und Erklärungen lassen sich problemlos in die Perl-Syntax umsetzen.

Gerade SVG eignet sich zur Ausgabe von Diagrammen aufgrund der hohen Skalierbarkeit einer SVG-basierten Grafik. Ich möchte an dieser Stelle auch an die Ausgabe eines Kreisdiagramms mit Hilfe der GD-Library anknüpfen. Wie bereits erwähnt, reicht es bei der GD-Funktion `imageArc` aus, lediglich den Start- und den Endwinkel anzugeben. Bei SVG sind jedoch der Start- und der Endpunkt erforderlich. Diese Punkte lassen sich mit ein wenig Trigonometrie berechnen. Alles, was dann dafür gebraucht wird, sind der Radius und ein Winkel, der sich aus der Gradzahl ergibt.

Vereinfachte Mathematik

Da ich an dieser Stelle nicht in reine Mathematik verfallen möchte, werde ich Ihnen eine vereinfachte Variante darstellen. Sehen Sie sich dazu die Abbildung 58.17 an.

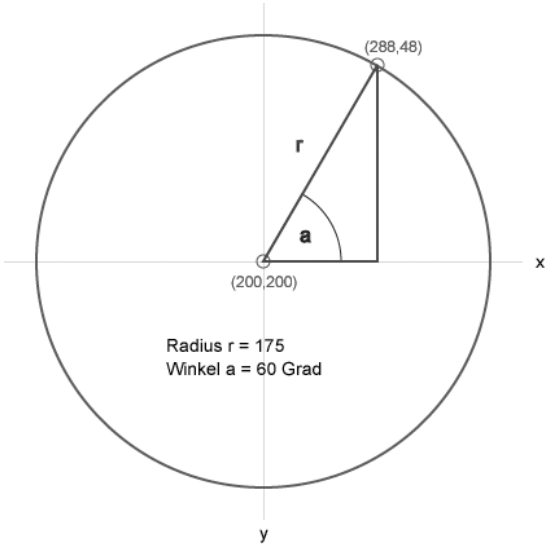

Abbildung 58.17 Schematische Darstellung zur Berechnung von Punkten auf einem Kreis

Wichtig zur Berechnung der Punkte auf der Kreisbahn ist der Winkel a und der Radius des Kreises r. Für jeden beliebigen Punkt auf der Kreisbahn lässt sich ein rechtwinkliges Dreieck zeichnen. Die längste Seite des Dreiecks, also die Gerade, die den Mittelpunkt mit dem Punkt auf der Kreisbahn verbindet, ist die Hypotenuse. Sie entspricht dem Radius des Kreises. Mit Sinus und Kosinus lässt sich nun die Verschiebung des Mittelpunktes auf der x- und auf der y-Achse berechnen.

Die Formel

$$x = cos(a) \times r$$

berechnet die Verschiebung auf der x-Achse und die Formel

$$y = sin(a) \times r$$

die Verschiebung auf der y-Achse. Der Winkel a hängt vom Start- und Endwinkel des Kreisbogens ab und bildet sich aus der Differenz dieser beiden Winkel. Der Startwinkel des Kreisbogens beträgt 30 und der Endwinkel 90. Somit ist der Winkel a = 60. Wenn Sie nun den Winkel a = 60 und den Radius r = 175 sowohl in die Formel zum Berechnen von x als auch von y einsetzen, kommen Sie zu folgenden Ergebnissen:

$$x = cos(60) \times 175 = 0{,}5 \times 175 = 87{,}5$$
$$y = sin(60) \times 175 = 0{,}866 \times 175 = 151{,}5$$

Grad und Bogenmaß

Abhängig davon, in welchem Quadranten des Kreises der Punkt liegt, müssen Sie x bzw. y addieren oder subtrahieren. Die Überprüfung des Quadranten selbst ist kein größeres Problem. Liegt der Winkel zwischen 0 und 90, ist es der erste bzw. linke untere Quadrant, liegt der Winkel zwischen 90 und 180; ist es der zweite bzw. rechte untere Quadrant usw. Spezialfälle sind jedoch die Winkel 0, 90, 180, 270 und 360. Der Kosinus dieser Winkel ist immer 0, und alles, was mit 0 multipliziert wird, ergibt ebenfalls 0. Diese Bedingungs- bzw. Werteprüfung ufert sehr schnell aus. PHP kann jedoch Abhilfe schaffen. Es hält nämlich eine Funktion bereit, die Gradangaben in das Bogenmaß umrechnet. Dabei können auch negative Werte zurückgegeben werden, und selbst bei Winkeln wie 0, 90, 180, 270 und 360 ergibt der Kosinus der Winkel nicht 0, sondern 1. Diese Funktion ist `deg2rad`.

```
double deg2rad(double)
```

Die Funktion wandelt die übergebene Gradzahl in das Bogenmaß um. Den Rückgabewert können Sie an die Funktionen `cos` oder `sin` übergeben. Sie erhalten dann immer einen Wert, den Sie nur addieren müssen.

Das Script aus Listing 58.14 nimmt Werte entgegen, die mit der POST-Methode von einem HTML-Formular übermittelt wurden, und zeichnet dazu passend ein Kreisdiagramm im SVG-Format.

```
<?php
  /* Variablen definieren */
  $cx = $_POST['cx'];
  $cy = $_POST['cy'];
  $cr = $_POST['cr'];
  $values = $_POST['value'];
  $descriptions = $_POST['description'];
```

```
  $colors = $_POST['color'];

  /* Gesamtsumme bilden */
  $value_sum = 0;
  for($i=0; $i<count($values); $i++)
  {
    $value_sum += $values[$i];
  }

  /* Header und SVG-Kopf ausgeben */
  header('Content-Type: image/svg+xml');
  echo "<?xml version=\"1.0\"?>\n";
  echo "<!DOCTYPE svg PUBLIC \"-//W3C//DTD SVG 1.0//EN\" \"http://
www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd\">\n\n";
  echo "<svg xmlns=\"http://www.w3.org/2000/svg\">\n";

  /* Start- und Endwinkel setzen */
  $ang_start = 270;
  $ang_end = 270;

  /* Kreissegmente ausgeben */
  for($i=0; $i<count($values); $i++)
  {
    // Berechnung der Winkel
    $ang_start = $ang_end;
    $ang_part = $values[$i] / $value_sum * 360;
    $ang_end = $ang_start + $ang_part;

    // Berechnung des Startpunkts auf der Kreisbahn
    $psx = round(cos(deg2rad($ang_start)) * $cr + $cx);
    $psy = round(sin(deg2rad($ang_start)) * $cr + $cy);

    // Berechnung des Endpunkts auf der Kreisbahn
    $pex = round(cos(deg2rad($ang_end)) * $cr + $cx);
    $pey = round(sin(deg2rad($ang_end)) * $cr + $cy);

    // Ausgabe des Kreissegments in SVG-Notation
    echo "  <path d=\"M $cx,$cy L $psx,$psy A $cr,$cr 0
             0,1 $pex,$pey Z\" fill=\"$colors[$i]\"
             stroke=\"black\" stroke-width=\"2\"/>\n";
  }

  /* Startpunkte festlegen */
  $desc_x = $cx + $cr + 50;
  $desc_y = 50;
```

```
  /* Bezeichnungen ausgeben */
  for($i=0; $i<count($descriptions); $i++)
  {
    echo "  <text x=\"$desc_x\" y=\"$desc_y\" font-size=\
             "14\" fill=\"$colors[$i]\">$descriptions[$i]
             - $values[$i]</text>\n";
    $desc_y += 18;
  }

  /* SVG-Fuss ausgeben */
  echo '</svg>';
?>
```

Listing 58.14 Zeichnen eines Kreisdiagramms im SVG-Format mit PHP

Das PHP-Script aus Listing 58.14 erwartet verschiedene Parameter vom aufrufenden HTML-Formular. Aufgrund der Länge des HTML-Formulars verzichte in an dieser Stelle auf eine Darstellung. Sie können es jedoch auf der dem Buch beiliegenden CD-ROM im Verzeichnis **x:\listings\dyngraph\list3.12.htm** finden.

Daten übergeben

Die ersten drei Variablen `$cx`, `$cy` und `$cr` erhalten den Mittelpunkt des Kreises und den Durchmesser. Zu den einzelnen Kreissegmenten werden sowohl eine Bezeichnung als auch ein Wert und eine Farbe übergeben, in der das Segment dargestellt werden soll. Diese werden jeweils als Array übergeben. Die Werte werden im Array `$values`, die Bezeichnungen im Array `$descriptions` und die Farben im Array `$color` gespeichert.

Im nächsten Schritt wird die Summe der einzelnen Werte des Arrays `$values` gebildet und in der Variablen `$value_sum` gespeichert. Dieser Wert wird benötigt, um später die Winkel der Kreisbögen zu errechnen.

Ausgabe im Browser

Dann wird der konstante Teil des SVG-Dokuments ausgegeben: zuerst der MIME-Typ des Dokuments (`image/svg+xml`) und anschließend die XML-Processing-Instruction, die DTD und das Start-Tag des Wurzelelements `svg`.

Danach werden der Startwinkel `$ang_start` und der Endwinkel `$ang_end` festgelegt. Beide werden zu Beginn auf den Wert 270 gesetzt. Dies führt dazu, dass mit dem Zeichnen der Kreissegmente nicht bei 3 Uhr, sondern bei 12 Uhr, also oben, begonnen wird.

Berechnung der Start- und Endpunkte

Die nun folgende `for`-Schleife arbeitet nun jeden Wert des Arrays `$values` einzeln ab, erzeugt den entsprechenden Pfad und sendet das Ganze mit einer `echo`-Anweisung an den Browser. Diese Schleife wird durchlaufen, solange Elemente im Array `$values` enthalten sind. Zu Beginn jedes Schleifendurchlaufs wird der Startwinkel auf den Endwinkel gesetzt. Der Grund ist, dass Sie somit pro Durchlauf immer nur den

neuen Endwinkel berechnen müssen, da der letzte Endwinkel dem neuen Startwinkel entspricht. Dann wird der neue Endwinkel berechnet. Dafür wird ein ganz normaler Dreisatz verwendet, denn die Summe aller Werte des Arrays `$values` verhält sich zum 360°-Winkel wie der aktuelle Wert zum Winkel a. Der Winkel a wird in diesem Fall dann in der Variablen `$ang_part` gespeichert. Das Ergebnis wird dann zu `$ang_start` addiert und anschließend in `$ang_end` gespeichert. `$ang_end` enthält nun den neuen Endwinkel. Im nächsten Schritt erfolgt die Berechnung des Start- und Endpunkts. Die Koordinaten x und y der Punkte werden dabei getrennt berechnet, da einmal Sinus und einmal Kosinus verwendet werden muss. Die Winkel werden dabei in das Bogenmaß umgewandelt und an die Funktion `sin` bzw. `cos` übergeben. Das Ergebnis wird dann mit dem Radius multipliziert, und es wird entweder `$cx` oder `$cy` hinzuaddiert.

```
$psx = round(cos(deg2rad($ang_start)) * $cr + $cx);
$psy = round(sin(deg2rad($ang_start)) * $cr + $cy);
$pex = round(cos(deg2rad($ang_end)) * $cr + $cx);
$pey = round(sin(deg2rad($ang_end)) * $cr + $cy);
```

`$psx` und `$psy` enthalten nun die Koordinaten für den Startpunkt und `$pex` und `$pey` die Koordinaten für den Endpunkt des Kreissegments. Am Ende der Schleife werden dann die errechneten Koordinaten in das vorgefertigte `path`-Konstrukt eingefügt und ausgegeben.

Legende ausgeben

Der nun folgende Schritt gibt lediglich die Bezeichnungen der Kreissegmente mit Werten in der entsprechenden Farbe unterhalb des Kreises aus. Am Ende des Scripts wird dann noch das Ende-Tag des `svg`-Elements ausgegeben.

Beispiel

In Abbildung 58.18 können Sie die Ausgabe des Scripts mit den Werten aus Tabelle 58.1 sehen. Der Mittelpunkt des Kreises wurde auf (200,200) festgesetzt und der Radius auf 175.

Bezeichnung	Wert	Farbe
Segment a	355	#CC0000
Segment b	342	#00CC00
Segment c	782	#0000CC
Segment d	453	#CCCC00
Segment e	535	#CC00CC
Segment f	333	#00CCCC

Tabelle 58.1 Werte für das Kreisdiagramm aus Abbildung 58.18

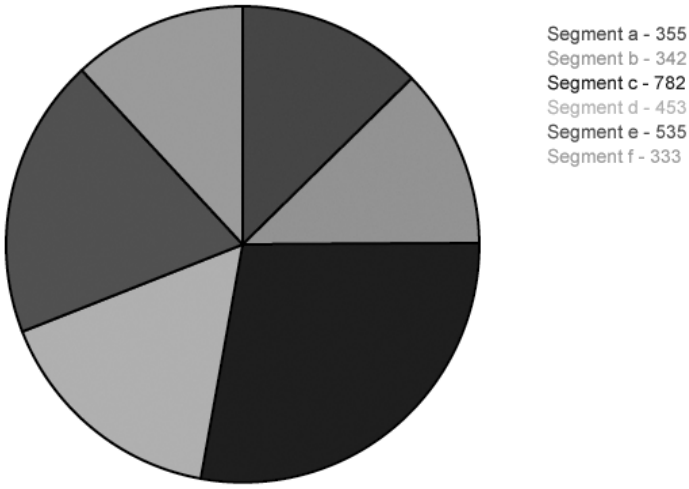

Abbildung 58.18 Ausgabe des PHP-Scripts aus Listing 58.14 mit den Werten aus der Tabelle 58.1

58.8 Zusammenfassung

- SVG ist ein auf Vektoren basierendes Grafik-Format und insbesondere zum Einsatz im Internet entwickelt worden.
- SVG wurde mit XML definiert und unterliegt den gleichen strikten Regeln.
- Texte werden mit dem `text`-Element ausgegeben.
- Grundformen wie Linien, Kreise und Rechtecke werden mit den Elementen `line`, `circle` und `rect` ausgegeben.
- Mit dem Attribut `fill` können Sie eine Füllfarbe für eine Form bestimmen und mit `stroke` und `stroke-width` eine Rahmenfarbe und -dicke.
- `linearGradient` und `radialGradient` ermöglichen einen linearen oder radialen Farbverlauf. Mit `stop`-Elementen werden die einzelnen Farben des Verlaufs definiert.
- Pfade ermöglichen weitaus komplexere Formen als Linien, Rechtecke und Kreise. Dadurch sind auch Vielecke oder Kreissegmente möglich. Selbst Bézier-Kurven lassen sich schnell und einfach definieren.
- SVG-Grafiken können sowohl mit PHP als auch mit Perl dynamisch erzeugt werden.

58.9 Fragen und Übungen

1. Wie lautet der MIME-Typ einer SVG-Grafik?
2. Mit welchem Element lässt sich eine SVG-Grafik in ein HTML-Dokument einbinden?
3. Definieren Sie einen linearen Farbverlauf mit der Farbe `#FFCC99` bei 0 % und `#663300` bei 100 %. Zeichnen Sie ein Rechteck, das den definierten Farbverlauf als Füllfarbe erhält.
4. Zeichnen Sie einen beliebigen Kreissektor, und füllen Sie ihn mit einem Farbverlauf Ihrer Wahl.
5. Setzen Sie das PHP-Script aus Listing 58.14 dieses Kapitels in ein Perl-Script um. Dieses Perl-Script soll ebenfalls Daten von einem HTML-Formular entgegennehmen und darauf aufbauend ein Kreisdiagramm in SVG erzeugen. Beachten Sie dabei Folgendes: Im HTML-Dokument müssen Sie bei den Werten für die `name`-Attribute der Eingabefelder die eckigen Klammern `[` und `]` entfernen. Außerdem kennt Perl die Funktion `round` nicht. Ignorieren Sie diese, da Sie in Perl keine ähnliche Funktion mit der gleichen Aufgabe benötigen. Außerdem kennt Perl keine Funktion, um Grad in Bogenmaß umzurechnen. Eine entsprechende Subroutine müssen Sie selbst definieren. Diese lautet:

```
sub deg2rad
{
  my $deg = shift;
  $rad = ($deg * 3.1415926535897932) / 180;
  return $rad;
}
```

59 RDF/RSS

Auch schlechte Nachrichten können wertvoll sein – wenn man sie früher hat als die anderen.
– Peter Hohl, dt. Journalist und Verleger

Content-Syndication, das ist das Schlagwort, mit dem das Einbinden von News oder anderen Inhalten von fremden Seiten in die eigenen bezeichnet wird. Dass dies durchaus von einigen Anbietern gewünscht ist, mag man bei dem Begriff eher weniger vermuten.

59.1 Was ist RDF/RSS?

Es gibt zahllose Quellen im Internet, durch die man an die neuesten Infos zu jedem Thema herankommt. Es kann jedoch gelegentlich ein wenig mühselig sein, nacheinander alle einschlägigen Seiten abzuklappern und nach einer halben Stunde wieder von vorn anzufangen, weil sich in der Zwischenzeit wieder einiges ereignet hat.

Parsen von HTML

Anfangs gab es einige findige Programmierer, die sich den HTML-Quelltext der Webseiten angesehen haben und exakt die Stellen herausgefiltert haben, an denen die News standen. Alle für sie interessanten Neuigkeiten konnten sie so bequem zusammenfassen und sogar in ihre eigenen Webseiten einbinden.

Die Zeiten dieser aufwendigen Frickelei sind aber schon ein ganze Weile vorbei. Mittlerweile haben sich Formate etabliert, mit denen sich News gewollt und vor allem einfach verteilen lassen. Eines dieser Formate ist RSS. Es wurde ursprünglich von Netscape entwickelt, um einen Austausch von News zwischen Portalen zu ermöglichen. In der Version 0.90 war dieses Format überaus mächtig, aber auch überladen. Es wurde kurzerhand vereinfacht und ist noch immer in der Version 0.91 aktuell im Netz in Verwendung.

RSS 1.0 und RDF

Nachdem Netscapes Interesse an Portalen stark absank und nicht mehr weiterentwickelt wurde, hat UserLand Software sich des Formats angenommen und es als Basis für seine Blog-Software genutzt. UserLand Software entwickelte das RSS-0.91-Format weiter und veröffentlichte die Versionen 0.92 bis 0.94, bis zur aktuellsten Version 2.0.

Parallel zu dieser Entwicklung durch UserLand blieb der alte Standard 0.90 erhalten, nur dass er als Version 1.0 »vermarktet« wurde. Dieser basiert nun vollständig auf RDF (Resource Description Framework). RDF ist ein durch das W3C standardisiertes Format, um Inhalte beschreiben zu können. Das Ziel von RDF ist die Entwicklung eines semantischen Webs, d.h., Inhalte werden durch Tags ausgezeichnet, die den Inhalt exakt bezeichnen, wodurch dieser eingeordnet werden kann (siehe Abschnitt 56.3.1).

Versions-Wahl

Um ein wenig Ordnung in diese unterschiedlichen Versionen zu bringen, werfen Sie einmal einen Blick auf die nachfolgende Tabelle.

Version	Entwickler	Bemerkung
0.90	Netscape	Ursprünglichste Form von RSS
0.91	UserLand	Die am einfachsten zu handhabende Version
0.92, 0.93, 0.94	UserLand	Weiterentwicklungen von RSS 0.91, die mehr Inhalte ermöglichen; Ablösung durch RSS 2.0
1.0	RSS-DEV Working Group	Basiert auf RDF, modular erweiterbar
2.0	UserLand	Durch Module erweiterbar

Tabelle 59.1 Die RSS-Versionen im Überblick

In der Regel treffen Sie im Internet auf die RSS-Version 0.91; und selbst wenn Ihnen einmal die Version 2.0 unterkommen sollte, beschränkt diese sich auf die von Version 0.91 bekannten Elemente.

59.2 RDF/RSS-Dokumente einsetzen

Ein Beispiel für ein RSS-Dokument könnte so aussehen (die Links im Dokument existieren übrigens nicht):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="0.91">
  <channel>
    <title>Mark-Lubkowitz.De</title>
    <description>Private Webseite des Autors</description>
    <link>http://www.mark-lubkowitz.de/</link>
    <language>de</language>
    <item>
      <title>Einstieg in MySQL erschienen</title>
      <link>http://www.mark-lubkowitz.de/news.php?id=5</link>
      <description>Das Werk zu MySQL für alle Einsteiger ist nun
        im Handel erhältlich.</description>
      <comments>http://www.mark-lubkowitz.de/news.php?id=5</comments>
      <pubDate>Thu, 29 May 2004 11:24:00 +0000</pubDate>
    </item>
    <item>
      <title>Relaunch von Mark-Lubkowitz.De</title>
      <link>http://www.mark-lubkowitz.de/news.php?id=4</link>
      <description>Ende des Jahres steht der erneute Relaunch der
        Webseite an.</description>
      <comments>http://www.mark-lubkowitz.de/news.php?id=4</comments>
      <pubDate>Thu, 29 May 2004 11:12:00 +0000</pubDate>
```

```
    </item>
  </channel>
</rss>
```

Listing 59.1 Beispiel für ein RSS-Dokument der Version 0.91

Schon bei einem unwissenden Blick auf das Beispiel aus dem vorangegangenen Listing lässt sich die Art und Weise, wie die News ausgetauscht werden, sehr schnell erkennen.

Wurzelknoten

Der Wurzelknoten in einem RSS-Dokument ist das Element `rss`. Die Version, auf der das RSS-Dokument beruht, wird mit dem Attribut `version` festgelegt.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="0.91">
  ...
</rss>
```

News-Kanal

Das zweitwichtigste Element eines RSS-Dokuments ist `channel`. Viele Webseiten unterteilen ihre News nach Kategorien. Normalerweise werden alle News, auch wenn sie einer unterschiedlichen Kategorie angehören, ein und demselben Channel zugeordnet. Es ist jedoch durchaus möglich, dass ein Betreiber einer Webseite für jede News-Kategorie einen eigenen Channel in das RSS-Dokument einfügt. Egal, ob nun ein oder mehrere Channels in einem RSS-Dokument enthalten sind, werden Sie in der Regel folgende Elemente innerhalb eines `channel`-Elements antreffen:

- `title`
  Der Titel des Channels. Für den Fall, dass es nur einen Channel gibt, ist es der Titel der Ursprungswebseite.
- `description`
  Eine Beschreibung zum Channel oder zur Webseite, von der die News stammen.
- `link`
  Entweder ein direkter Link zu der Newskategorie oder zur Ursprungswebseite.
- `language`
  Die Sprache, in der die News verfasst wurden. Zur Angabe, welche Sprache verwendet wird, kommen die Sprachkürzel zum Einsatz.

Alle diese Elemente dürfen nur einmal pro Channel verwendet werden, sollten aber auf jeden Fall eingesetzt werden.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="0.91">
  <channel>
    <!-- Infos zum Channel oder zur Webseite -->
    <title>Mustermann.De</title>
```

```
    <description>Webseite von Max Mustermann</description>
    <link>http://www.mustermann.de/</link>
    <language>de</language>
    ...
  </channel>
</rss>
```

Item Die eigentlichen News, die in einem RSS-Dokument enthalten sind, werden mit den `item`-Elementen definiert, die Kindelemente der `channel`-Elemente sind.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="0.91">
  <channel>
    <!-- Infos zum Channel oder zur Webseite -->
    <title>Mustermann.De</title>
    <description>Webseite von Max Mustermann</description>
    <link>http://www.mustermann.de/</link>
    <language>de</language>
    <!-- Hier folgen die News -->
    <item>
      ...
    </item>
    ...
  </channel>
</rss>
```

Folgende Elemente können als Kindelemente von `item` notiert werden:

- `title`
  Der Titel der Nachricht.
- `description`
  Ein kurzer Anreißer oder Teaser, der Interesse an der Nachricht weckt.
- `link`
  Ein direkter Link zu der Nachricht, über den diese *immer* erreichbar ist.
- `comments`
  Ein Link zu den Kommentaren zu dieser Nachricht, falls das Kommentieren auf der Webseite möglich ist. Andernfalls ist es häufig auch der Link direkt zur Nachricht.
- `pubDate`
  Das Datum, wann die Nachricht veröffentlicht wurde.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rss version="0.91">
  <channel>
    <!-- Infos zum Channel oder zur Webseite -->
    <title>Mustermann.De</title>
```

```
    <description>Webseite von Max Mustermann</description>
    <link>http://www.mustermann.de/</link>
    <language>de</language>
    <!-- Hier folgen die News -->
    <item>
      <title>Musternachricht</title>
      <description>Nachricht im RSS-Feed.</description>
      <link>http://www.mustermann.de/news.php?id=3</link>
      <comments>http://www.mustermann.de/newscomments.php
        ?id=3</comments>
      <pubDate>Wed, 25 Aug 2004 18:01:00 +0000</pubDate>
    </item>
  </channel>
</rss>
```

Listing 59.2 Vollständiges Beispiel für ein RSS-Dokuments

Wie bereits zuvor erwähnt, können mehrere `item`-Elemente unterhalb eines `channel`-Elements notiert werden. Jedoch sollte dies nicht ausarten. In der Regel werden 10 bis 20 News in einem RSS-Feed veröffentlicht.

Die Informationen in diesem Kapitel spiegeln den kleinsten gemeinsamen Nenner wider und sind kompatibel zu den Version 0.91 und 2.0.

59.3 Zusammenfassung

- RSS bezeichnet ein auf XML basierendes Format, mit dem News ausgetauscht werden können.
- RDF ist eine Technologie, um Inhalte semantisch korrekt in eine Beziehung zu bringen.
- Die am häufigsten anzutreffenden Formate sind RSS 0.91, RSS 2.0 und RDF.
- Channels dienen dazu, Nachrichten in einem Newsfeed nach Kategorien unterteilen zu können. Nur wenige Seiten nutzen die Möglichkeiten der Channel in ihren Newsfeeds.

59.4 Fragen und Übungen

1. Gibt es Unterschiede zwischen Newsfeeds im Format RSS und RDF?
2. Welches Attribut müssen Sie im Start-Tag des `rss`-Elements unbedingt angeben?
3. Welches Element steht bei einem RSS-Dokument für den Titel einer Nachricht?
4. Welche Information wird in einem RSS-Dokument bei einer Nachricht mit dem Element `comments` ausgezeichnet?
5. Müssen Channels in einem RSS-Dokument verwendet werden?

60 Webservices

Wer uns vor nutzlosen Wegen warnt, leistet uns einen ebenso guten Dienst, wie derjenige, der uns den rechten Weg anzeigt.
– Heinreich Heine, (1797–1856), dt. Dichter

Kommunikation ist wichtiger denn je. Wir Menschen bedienen uns dazu der Telefone, E-Mails, Faxgeräte, der Post und der Instant Messenger, um Informationen auszutauschen. Aber wie kommunizieren eigentlich Anwendungen, die auf unterschiedlichen Systemen laufen? Ganz einfach: durch Webservices.

60.1 Was sind Webservices?

Grundsätzlich stellt die Kommunikation zwischen zwei Computern kein Problem dar, denn es gibt Netzwerke. Jedoch ist die Kommunikation zwischen zwei Computern in einem Netzwerk häufig eher von der Art Mensch-zu-Maschine. Der Benutzer gibt im Browser eine URL ein, dieser ruft die Daten vom Server ab und präsentiert sie dem Benutzer. Webservices dienen jedoch z.B. dem Informationsfluss zwischen zwei Servern. Ein Beispiel: Server A fragt bei Server B nach einer bestimmten Information wie der Uhrzeit oder seiner Festplattengröße. Server B überprüft, ob er diese Information zur Verfügung stellen kann, und antwortet dem Server A entsprechend.

Grundsätzlich klingt das sehr einfach, aber nicht alle Server sind gleich. Sie unterscheiden sich z.B. im Betriebssystem, der installierten Hardware oder den Anwendungen, die den Informationsaustausch durchführen sollen. Es muss also eine gemeinsame Sprache gefunden werden, die von allen verstanden und »gesprochen« werden kann. Auch die Weltbevölkerung hat dieses Problem, weshalb irgendwann Englisch als die internationale Sprache festgelegt wurde. Außerdem muss diese »Computersprache« auch in der Lage sein, unabhängig von der Art der auszutauschen Informationen zu sein.

60.2 Die unterschiedlichen Varianten

Mittlerweile haben sich einige Standards für Webservices etabliert, die ich Ihnen nachfolgend kurz vorstellen werde.

60.2.1 XML-RPC

Dieser Standard basiert auf den so genannten »**R**emote **P**rocedure **C**alls« (RPCs). RPC bedeutet, dass die Anfrage so formuliert wird, dass auf dem entfernten System eine Funktion aufgerufen wird. Dieser Funktion können Parameter übergeben werden –

also fast genauso, als würden Sie eine Funktion in einer Programmiersprachen aufrufen, nur dass dabei XML genutzt wird, um den Aufruf selbst zu formulieren.

Ein Beispiel:

```
<?xml version="1.0"?>
<methodCall>
  <methodName>translate</methodName>
  <params>
    <param><string>Hallo Welt!</string></param>
  </params>
</methodCall>
```

Listing 60.1 Beispiel für einen XML-RPC

Erklärung In diesem Beispiel wird auf einem entfernten Server die Funktion `translate` aufgerufen. Dies wird durch das Element `methodName` festlegt. Als Parameter wird ihr die Zeichenfolge `Hallo Welt!` übergeben, festgelegt durch das Element `param`. Interessant ist, dass eine Typisierung des Wertes erfolgt, indem das `string`-Element genutzt wird. Da es in einem XML-Dokument nur Zeichenfolgen gibt, kann so zwischen normalem Text, Zahlen und anderen Datentypen unterschieden werden.

HTTP Anfragen, die mit XML-RPC gestellt werden, erfolgen über das HTTP-Protokoll. Der Server, an den die Anfrage gerichtet wird, muss also nicht erst ein neues Protokoll erlernen, sondern es reicht aus, wenn auf ihm ein HTTP-Daemon installiert ist, der diese Anfragen verarbeiten kann.

60.2.2 SOAP

SOAP, das »Simple Object Access Protocol«, beschreitet einen ähnlichen Weg wie XML-RPC. Es ist trotz des anders lautenden Namens jedoch nicht trivial, einen Webservice mit SOAP zu erstellen. Zusammenfassend lässt sich sagen, dass SOAP alle Nachrichten in einem Umschlag (engl. *envelope*) verpackt. Das Besondere an diesem Umschlag ist, dass er beschreibt, welche Daten in der Nachricht enthalten sind und von wem und vor allem wie diese Daten zu verarbeiten sind.

Nachricht Der Unterschied zu einem RPC ist, dass Nachrichten keine Antworten erwarten. Wenn Sie z. B. mittels XML-RPC auf einem entfernten System eine Funktion aufrufen, erwarten Sie in der Regel eine Antwort, z. B. einen Rückgabewert oder Ähnliches. Nachrichten bedürfen keiner Antwort. Dies ähnelt einem Telefongespräch und einer E-Mail. Wenn Sie jemanden anrufen, möchten Sie zeitnah eine Antwort oder Reaktion. Eine E-Mail hingegen schicken Sie ab, erledigen in der Zwischenzeit andere Dinge und bekommen irgendwann eine Antwort oder auch nicht.

Ein Beispiel für Frage und Antwort:

> Frage: Kannst du mir »Hallo Welt!« ins Englische übersetzen?
>
> Antwort: Ja, es heißt »Hello world!«

Ein Beispiel für eine Nachricht:

> Nachricht: Übersetze mir »Hallo Welt!« ins Englische, und schicke mir nach Möglichkeit eine Antwort.

Dies bedeutet jedoch nicht, dass RPC mit SOAP nicht möglich wäre, sondern dass es nur eine der verschiedenen Möglichkeiten ist.

Eine SOAP-Nachricht könnte folgendermaßen aussehen:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
">
  <soap:Body>
    <translate xmlns="http://www.translator.tld/dictionary">
      <text>Hallo Welt!</text>
    </translate>
  <soap:Body>
</soap:Envelope>
```

Listing 60.2 Beispiel für eine SOAP-Nachricht

Umschlag
Namespace

Zunächst folgen der Umschlag (`soap:Envelope`) und der Namespace für SOAP (Attribut `xmlns`). Einen Namespace können Sie sich in etwa wie eine Bibliothek vorstellen. Bei PHP wäre so etwas eine Datei, die Sie über `include` einbinden und in der Funktionen, Variablen oder auch Klassen definiert worden sind. Diese sind anschließend in dem PHP-Dokument verfügbar.

Anschließend folgt die eigentliche Nachricht, und zwar im `soap:Body`-Element:

```
<soap:Body>
  <translate xmlns="http://www.translator.tld/dictionary">
    <text>Hallo Welt!</text>
  </translate>
<soap:Body>
```

Auch hier wird wieder ein Namespace eingebunden, der festlegt, wo `translate` zu finden ist. Anschließend folgen Werte für `translate`.

So kompliziert, wie es auf den ersten Blick wirkt, ist SOAP auch. Ich kann Ihnen nur das Buch »Web Services – Die Standards« von Tobias Hauser und Ulrich M. Löwer, erschienen bei Galileo Press (ISBN: 3–89842–393-X), empfehlen, falls Sie ein tiefergehendes Interesse an SOAP und dessen Entwicklung haben. Ansonsten sollten Sie die wesentlich einfachere Variante XML-RPC im Hinterkopf behalten, denn auf diese werden wir etwas später noch einmal zurückkommen.

60.2.3 WSDL

Mit WSDL wird eine Möglichkeit bezeichnet, einen Webservice zu beschreiben, daher auch der Name: **W**eb **S**ervice **D**escription **L**anguage. SOAP ist z. B. so umfangreich und flexibel, dass Sie wissen müssen, welche Methoden ein SOAP-Service bietet, und diese Definition erfolgt in den meisten Fällen mit WSDL. Stellen Sie sich SOAP einfach als den Wortschatz einer Sprache vor, also als die Vokabeln. Nun benötigen Sie jedoch Regeln, um diese Wörter in eine sinnvolle Reihenfolge zu bringen, und das ist die Grammatik bzw. WSDL.

Vokabeln lassen sich sehr schnell erklären und lernen, aber mit der Grammatik ist das immer so eine Sache. Selbst eine einfache Erklärung für den kleinsten gemeinsamen Nenner in einer Sprache führt zu einer komplexen Struktur. Aus diesem Grunde verzichte ich an dieser Stelle auch auf ein Beispiel, weil dies fast eine Seite füllen würde.

60.3 Webservices nutzen

In letzter Zeit wird die eBay-API immer beliebter. Sie hat sich zwar noch keinem gängigen Standard untergeordnet, mit ihr lässt sich aber die Funktionsweise von Webservices sehr schnell und übersichtlich verdeutlichen, da sie dem XML-RPC-Standard sehr stark ähnelt. In den folgenden Unterkapiteln werden zum einen die Maßnahmen zur Registrierung beim eBay Developer Program vorgestellt, und zum anderen wird erläutert, wie Sie mit PHP die aktuelle Uhrzeit des eBay-Servers »erfragen« können.

60.3.1 eBay Developer Program

Bevor Sie Zugriff auf die eBay-API erhalten können, ist eine Registrierung beim eBay Developer Program notwendig. Dieses finden Sie unter der Adresse:

http://developer.ebay.com

Anmeldung

Wählen Sie auf der Startseite den Link **Membership · Join** aus (siehe Abbildung 60.1 links unten). Auf der sich öffnenden Seite können Sie mit der Registrierungsprozedur beginnen. Sie erhalten einige Informationen darüber, wie die einzelnen Schritte aussehen. Wählen Sie **Join now**. Auf der ersten Seite müssen Sie Informationen über sich eingeben sowie etwaige Firmenkontaktdaten. Klicken Sie nach der Eingabe dieser Daten auf den Button **Next** am unteren Seitenende.

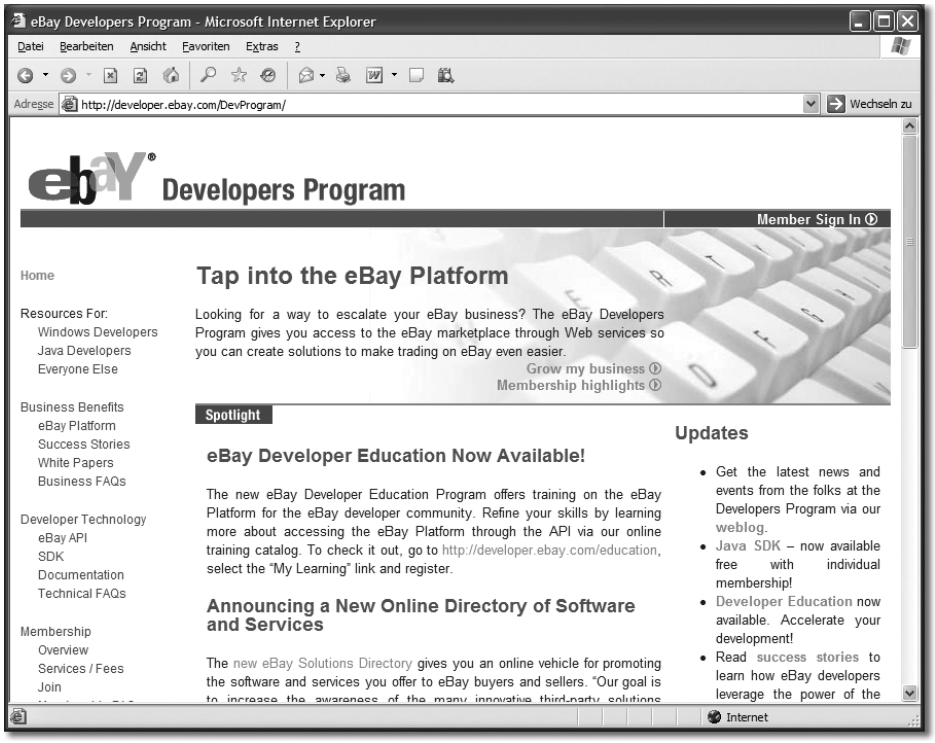

Abbildung 60.1 Startseite des eBay Developer Program

Lizenz lesen

Nun wird die API-Lizenz angezeigt. Für Testzwecke und zum Nachvollziehen der Beispiele in diesem Kapitel reicht die *Individual API License* vollkommen aus. Lesen Sie sich die Bedingungen dieser Lizenz in Ruhe durch, und klicken Sie erneut auf **Next**.

Lizenz wählen

Nachdem Sie sich über die Lizenz informiert haben, können Sie diese nun auswählen. Die *Individual License* finden Sie gleich zu Beginn der Seite. Markieren Sie den Radiobutton **Individual Tier** sowohl im Bereich **API License Decision and Pricing Tier** als auch im Bereich **Developer Technical Support Tier**. Außerdem sollten Sie alle Auswahlmöglichkeiten im Bereich **Permitted Use** und **Terms and Conditions that Apply to the API License** lesen und die zutreffenden entsprechend markieren. Wechseln Sie mit **Next** zur nächsten Seite.

Vereinbarungen

Es folgen nun einige Vereinbarungen zwischen Ihnen und eBay. Lesen und bestätigen Sie diese durch das Aktivieren der Checkboxen. Wählen Sie abschließend **I Accept This Agreement**.

eBay-Account

Nun folgt die Eingabe der Daten Ihres eBay-Accounts. Das sind die gleichen Daten, die Sie auch nutzen, um bei ebay.de in Auktionen mitzubieten. Nach dem Abschluss der Registrierung erhalten Sie eine Bestätigungs-E-Mail von eBay. Diese E-Mail ist

außerordentlich wichtig, weil sie zum Erzeugen der Lizenzschlüssel, der Sandbox-Keys, benötigt wird. Folgen Sie dazu den Anweisungen und Punkt 1, 2 und 3 der E-Mail. Nachdem die Keys erzeugt worden sind, speichern Sie diese ab oder drucken sie aus.

Test-Account

Was Sie nun benötigen, ist ein Test-Account. Gehen Sie auf **http://sandbox.ebay.com/**, und registrieren Sie dort einen Account. Wichtig ist die Angabe einer gültigen Adresse in den USA. Um an eine solche Adresse zu gelangen, können Sie die Yellow Pages nutzen, die Sie unter **http://www.yellowpages.com/** finden.

Dieser »Hack« mit den Yellow Pages ist durchaus legal, solange Sie einen Testaccount einrichten. Da Sie bei der *Individual Tier-Lizenz* auf einem Testsystem arbeiten, nämlich in der so genannten »Sandbox«, ist dies erlaubt. Beim echten ebay.com wäre dies natürlich ein Verstoß gegen die Regeln. Den »Hack« können Sie übrigens auch im eBay Developer Forum finden.

Auth & Auth

Neben den durch die Registrierung erhaltenen Daten benötigen Sie noch einen *Auth & Auth Token*. Diesen können Sie unter der folgenden Adresse erhalten:

http://developer.ebay.com/tokentool/Credentials.aspx

Wählen Sie aus dem Dropdown-Menü **Sandbox** aus, und tragen Sie in die restlichen Felder Ihre Daten ein. Generieren Sie anschließend den Token durch einen Klick auf **Continue to generate token**, und speichern Sie diesen nach Möglichkeit in einer Datei ab.

Geschafft! Nun wird es Zeit für einen ersten Einsatz!

60.3.2 Grundlagen

Eine Anfrage an die eBay-API gliedert sich in zwei Stücke: den Header und die Anfrage.

Header

In den vorangegangenen Teilen dieses Buches haben Sie bereits MIME-Typen kennen gelernt und Sie wissen, wie Sie mit der PHP-Funktion `header` dem Browser einen anderen MIME-Typ mitteilen können. Vor allem bei der Erzeugung dynamischer Grafiken ist dies wichtig. Ein Beispiel:

```
header('Content-Type: image/png');
```

Modifizierter Header

Für den Aufruf einer Funktion der eBay-API sind jedoch noch einige weitere Informationen erforderlich, die im Header mitgeteilt werden müssen.

```
X-EBAY-API-COMPATIBILITY-LEVEL:
X-EBAY-API-SESSION-CERTIFICATE:
X-EBAY-API-DEV-NAME:
X-EBAY-API-APP-NAME:
X-EBAY-API-CERT-NAME:
X-EBAY-API-CALL-NAME:
X-EBAY-API-SITEID:
X-EBAY-API-DETAIL-LEVEL:
Content-Type:
Content-Length:
```

Listing 60.3 Beispiel für den modifizierten HTTP-Header

Versionskontrolle

Die Header-Information `X-EBAY-API-COMPATIBILITY-LEVEL` legt fest, auf welche Version der eBay-API sich diese Anfrage bezieht. Der Grund ist, dass sich die eBay-API im Laufe der Zeit weiterentwickelt und erweitert oder verändert wird. Jeder Kompatibilitätslevel bleibt für ca. 1 Jahr aktuell und abwärtskompatibel. Momentan ist dies der Level `367`.

Zugangsdaten

Für die drei Informationen `X-EBAY-API-DEV-NAME`, `...APP-NAME` und `...CERT-NAME` werden die Werte angegeben, die Sie bei der Registrierung erhalten haben. Diese Werte werden Sie noch häufiger benötigen und somit empfiehlt es sich, diese in eine externe Datei auszulagern.

> Achten Sie darauf, diese Zugangsdaten nicht an andere Personen weiterzugeben, da dies unter Umständen Kosten nach sich ziehen kann, wenn Sie eine andere Lizenz als die *Individual Tier*-Lizenz nutzen.

Zertifikat

Der Wert für `X-EBAY-API-SESSION-CERTIFICATE` basiert auf den zuvor genannten drei Sandbox-Keys und wird aus diesen folgendermaßen aufgebaut:

```
dev-name; app-name; cert-name
```

Die Werte werden also durch ein Semikolon getrennt aneinander gehängt.

Funktionsname

Der Name der Funktion, die Sie aufrufen möchten, wird mit Hilfe der Information `X-EBAY-CALL-NAME` angegeben. Für den Abruf der Server-Uhrzeit ist dies die Funktion `GeteBayOfficialTime`.

International

Da eBay eine internationale Plattform ist, gibt es für fast jedes Land ein eigenes eBay. *eBay.com* und *eBay.de* führen unterschiedliche Auktionen, und auch die Benutzer sind andere. Es muss also mit angegeben werden, für welche der eBay-Seiten Ihre Anfrage gültig ist. Dies geben Sie mit der Information `X-EBAY-API-SITEID` an. Für die USA lautet die ID z.B. 0 und für Deutschland 77.

Die IDs anderer Seiten finden Sie in der Dokumentation zur API unter

http://developer.ebay.com/DevZone/docs/API_Doc/

So lange Sie Ihre Programme jedoch in der Sandbox testen, beziehen sich die erhaltenen Informationen immer auf die amerikanische Plattform, weil die Sandbox eine Kopie von eBay.com ist.

Detail-Level

Die Information `X-EBAY-API-DETAIL-LEVEL` legt schließlich fest, wie detailliert die Antwort der eBay-API sein soll. Wird von der Funktion kein Detail-Level unterstützt, lautet der Standardwert 0.

Anfrage

Einen Teil der Informationen, die Sie bereits im HTTP-Header übergeben haben, müssen außerdem im XML-Dokument wiederholt werden. Ein Beispiel für die Anfrage:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<request>
  <RequestToken>SANDBOX_TOKEN</RequestToken>
  <RequestUserId>EBAY_TESTUSER_USERNAME</RequestUserId>
  <RequestPassword>EBAY_TESTUSER_PASSWORD</RequestPassword>
  <ErrorLevel>1</ErrorLevel>
  <DetailLevel>0</DetailLevel>
  <Verb>GeteBayOfficialTime</Verb>
  <SiteId>0</SiteId>
</request>
```

Listing 60.4 Beispiel für eine XML-Anfrage

Das Wurzelelement einer Anfrage ist immer `request`. Aus dem Header sind bereits die folgenden Informationen bekannt: `DetailLevel`, `SiteId` und auch `Verb` (alias `CALL-NAME`). Der Rest ist neu.

Token

Den aus den drei Sandbox-Keys generierten Token geben Sie mit dem Element `RequestToken` an. Dazu muss es lediglich zwischen den beiden Tags notiert werden.

Benutzername und Passwort

Den Benutzernamen und das Passwort des Users geben Sie mit `RequestUserId` und `RequestPassword` an. Diese Daten müssen zu einem existierenden eBay-User gehören, z.B. zu dem Test-Account, den Sie in der Sandbox eingerichtet haben.

ErrorLevel

Sollte es während der Ausführung seitens der eBay-API zu einem Fehler kommen, z.B. weil Ihre Anfrage falsch gestellt ist, gibt Ihnen die eBay-API eine Fehlermeldung zurück. Je nachdem, wie hoch der Level ist, wird die Meldung detaillierter oder ungenauer. Solange Sie in der Sandbox arbeiten und noch »üben«, sollten Sie den ErrorLevel 1 nutzen. Dieser gibt detailliert genug aus, warum es zu einem Fehler gekommen ist, und das Problem kann anschließend gelöst werden.

60.3.3 Beispiel

Die Umsetzung in PHP ist für einen Anfänger ein wenig knifflig. Dies liegt zum einen daran, dass Sie den Header modifizieren müssen, und zum anderen daran, dass die Anfragen verschlüsselt mit HTTPS erfolgen. Leider können Sie zum Modifizieren des Headers nicht auf die Funktion `header` zurückgreifen. Der Grund dafür ist, dass PHP-Scripts normalerweise auf Anfragen von einem Browser reagieren. Ändern Sie also den HTTP-Header mit `header`, wird dieser geänderte HTTP-Header an den Browser gesendet. Für den Zugriff auf die eBay-API muss aber in diesem Moment Ihr Script die Aufgabe des Browsers übernehmen.

cURL

Für solche Aufgaben gibt es jedoch eine Bibliothek für PHP, die diese Möglichkeiten zur Verfügung stellt: cURL. Mit Hilfe von cURL können Sie jede beliebige Art von Anfragen stellen, die auf dem HTTP-Protokoll basieren.

Standardmäßig ist cURL nicht aktiviert, so dass Sie nun PHP entsprechend konfigurieren müssen. Falls Sie Zugriff auf die **php.ini** haben, müssen Sie lediglich das Semikolon ; vor der Zeile

```
;extension=php_curl.dll
```

entfernen. Eine andere Möglichkeit wäre, die Bibliothek dynamisch zur Laufzeit des Scripts zu aktivieren. Hier brauchen Sie die Funktion `dl`. Je nach Betriebssystem, auf dem der Apache und PHP laufen, trägt die Bibliothek einen anderen Namen, weshalb sich auch der Aufruf unterscheidet.

Unter Windows:

```
dl('php_curl.dll');
```

Unter Linux:

```
dl('php_curl.so');
```

Konfigurationsdatei

Anstatt in jedem Script die Sandbox-Keys oder den Token erneut zu hinterlegen, sollte dafür eine Konfigurationsdatei verwendet werden, die lediglich in das eigentliche Script eingebunden wird:

```
<?php
  $config['compatibility_level'] = '367';
  $config['dev_name'] = 'DEV-NAME';
  $config['app_name'] = 'APP-NAME';
  $config['cert_name'] = 'CERT-NAME';
  $config['session_certificate'] = $config['dev_name'].';'.
    $config['app_name'].';'.$config['cert_name'];
  $config['request_token'] = 'TOKEN';
  $config['siteid'] = '0';
```

```
  $config['detaillevel'] = '0';
?>
```

Listing 60.5 Die Datei config.inc.php

Natürlich müssen Sie die Werte in der **config.inc.php** durch Ihre eigenen ersetzen. Dies betrifft in diesem Fall `dev_name`, `app_name`, `cert_name` und `request_token`.

Hauptscript Das Hauptscript sieht wie folgt aus:

```
<?php

  /* Konfigurationsdatei einbinden */
  include('config.inc.php');

  $ebay_username = 'EBAY_TESTUSER_USERNAME';
  $ebay_password = 'EBAY_TESTUSER_PASSWORD';
  $ebay_url = 'https://api.sandbox.ebay.com/ws/api.dll';

  /* XML-Dokument erzeugen */
  $apiXML = '<?xml version="1.0" encoding="iso-8859-1"?>';
  $apiXML .= '<request>';
  $apiXML .= '<RequestToken>'.$config['request_token']
             .'</RequestToken>';
  $apiXML .= '<RequestUserId>'.$ebay_username
             .'</RequestUserId>';
  $apiXML .= '<RequestPassword>'.$ebay_password
             .'</RequestPassword>';
  $apiXML .= '<ErrorLevel>1</ErrorLevel>';
  $apiXML .= '<DetailLevel>'.$config['detaillevel']
             .'</DetailLevel>';
  $apiXML .= '<Verb>GeteBayOfficialTime</Verb>';
  $apiXML .= '<SiteId>'.$config['siteid'].'</SiteId>';
  $apiXML .= '</request>';

  /* Header erzeugen */
  $apiHeader[] = 'X-EBAY-API-COMPATIBILITY-LEVEL: '
                 .$config['compatibility_level'];
  $apiHeader[] = 'X-EBAY-API-SESSION-CERTIFICATE: '
                 .$config['session_certificate'];
  $apiHeader[] = 'X-EBAY-API-DEV-NAME: '
                 .$config['dev_name'];
  $apiHeader[] = 'X-EBAY-API-APP-NAME: '
                 .$config['app_name'];
  $apiHeader[] = 'X-EBAY-API-CERT-NAME: '
                 .$config['cert_name'];
```

```
  $apiHeader[] = 'X-EBAY-API-CALL-NAME:
                  GetebayOfficialTime';
  $apiHeader[] = 'X-EBAY-API-SITEID: '.$config['siteid'];
  $apiHeader[] = 'X-EBAY-API-DETAIL-LEVEL: '
                 .$config['detaillevel'];
  $apiHeader[] = 'Content-Type: text/xml';
  $apiHeader[] = 'Content-Length: '.strlen($apiXML);


  /* Anfrage senden */
  $apiAnswer = '';
  $chApi = curl_init($ebay_url);
  if($chApi)
  {
    curl_setopt($chApi,CURLOPT_RETURNTRANSFER,1);
    curl_setopt($chApi,CURLOPT_POST,1);
    curl_setopt($chApi,CURLOPT_POSTFIELDS,$apiXML);
    curl_setopt($chApi,CURLOPT_HTTPHEADER,$apiHeader);
    $apiAnswer = curl_exec($chApi);
  }
  else
  {
    die('Fehler bei der Initialisierung von cURL.');
  }

  /* Antwort überprüfen */
  if($apiAnswer == false)
  {
    header('Content-Type: text/html');
    echo 'Fehler:<br><b>'.curl_errno($chApi).'</b> / '.curl_
error($chApi);
  }
  else
  {
    /* Daten ausgeben */
    header('Content-Type: text/xml');
    echo $apiAnswer;
  }

?>
```

Listing 60.6 Das eigentliche Script, das die Anfrage stellt

Zunächst erfolgt das Einbinden der Konfigurationsdatei **config.inc.php**. Da noch niemand anderes außer uns das Skript nutzen wird, werden der Benutzername und das Kennwort des eBay-Testusers direkt im Script festgelegt. In einem produktiven Einsatz

sollten diese Daten nach Möglichkeit über das Array `$_POST` entgegengenommen werden. Anschließend wird noch die URL zur eBay-API der Sandbox in der Variablen `$ebay_url` gespeichert.

XML-Dokument definieren

Da Sie die exakte Länge des XML-Dokuments in Byte benötigen, müssen Sie dieses definieren, bevor Sie den Header modifizieren können. Das eigentliche Dokument unterscheidet sich von dem aus Listing 60.4 nur darin, dass die korrekten Werte eingefügt werden. Das gesamte Dokument wird als Zeichenkette in der Variablen `$apiXML` gespeichert.

Header modifizieren

cURL erwartet, dass alle Zusatzangaben oder Modifizierungen der Informationen im Header als Array übergeben werden. Dabei wird für jede Information ein eigenes Element in dem Array verwendet. Jede Angabe wird außerdem als zusammenhängende Zeichenkette übergeben. Um die Länge des XML-Dokuments festlegen zu können, müssen Sie lediglich die Funktion `strlen` aufrufen und ihr als Parameter `$apiXML` übergeben:

```
$apiHeader[] = 'Content-Length: '.strlen($apiXML);
```

Anfrage ausführen

Nach diesen Vorbereitungen kann die Anfrage durchgeführt werden. cURL gibt Ihnen die Antwort als String zurück. Dieser String wird in der Variablen `$apiAnswer` gespeichert. Zunächst wird mit `curl_init` ein neues Handle erzeugt und der Variablen `$chApi` zugewiesen. Zusätzlich wird der Funktion die URL übergeben, zu der eine Verbindung hergestellt werden soll, in diesem Fall die eBay-API der Sandbox.

```
$chApi = curl_init($ebay_url);
```

Ist der Rückgabewert der Funktion ungleich `false`, war der Versuch erfolgreich. Im `if`-Anweisungsblock werden mit `curl_setopt` einige Informationen an cURL übergeben:

- `CURLOPT_RETURNTRANSFER`
  Ein Wert ungleich 0 bedeutet, dass die Antwort auf die Anfrage als String zurückgegeben wird. Andernfalls würde cURL direkt die Ausgabe vornehmen.
- `CURLOPT_POST`
  Ein Wert ungleich 0 bedeutet, dass die Daten mittels POST an den Empfänger verschickt werden sollen.
- `CURLOPT_POSTFIELDS`
  Die mittels POST zu übertragenden Daten werden durch diesen Parameter übergeben. Im Fall der Anfrage ist dies das eigentliche XML-Dokument, das in `$apiXML` gespeichert wurde.

- `CURLOPT_HTTPHEADER`
  Um den Header zu modifizieren, werden die Informationen mit diesem Parameter übergeben. Die notwendigen Modifizierungen wurden im Array `$apiHeader` gespeichert, das mit diesem Parameter übergeben wird.

Ausführung

Die Funktion `curl_exec` führt schlussendlich die Anfrage aus. Als Parameter muss das cURL-Handle übergeben werden. Je nachdem, ob cURL die Ausgabe der Antwort vornehmen soll, gibt die Funktion die Antwort als String zurück.

Sollte beim Versuch des Verbindungsaufbaus ein Fehler auftreten, wird die Ausführung des Scripts im `else`-Anweisungsblock abgebrochen.

Antwort ausgeben

Schlussendlich muss noch überprüft werden, ob überhaupt eine Antwort verfügbar ist. Ist der Wert von `$apiAnswer` gleich `false`, dann haben wir keine Antwort bekommen. In einem solchen Fall gibt `curl_errno` die Fehlernummer und `curl_error` die Fehlermeldung zurück.

Wurde eine Antwort geliefert, erfolgt ihre Ausgabe im `else`-Anweisungsblock. Normalerweise müsste nun eine Verarbeitung der Antwort erfolgen. Die Antwort ist immer ein XML-Dokument. In diesem Beispiel wird die Antwort jedoch einfach ausgegeben. Damit eine entsprechende Darstellung erfolgt, wird dem Browser mitgeteilt, dass es sich um ein XML-Dokument handelt.

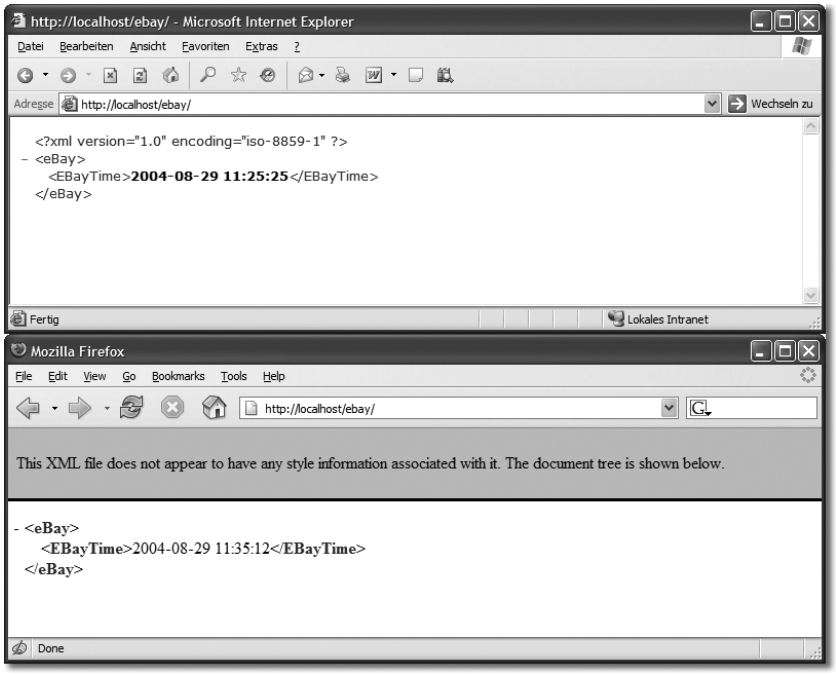

Abbildung 60.2 Die Ausgabe des Scripts im Internet Explorer 6.0 und im Mozilla Firefox 0.93

In Abbildung 60.2 können Sie die Antwort der eBay-API auf unsere Frage nach der Uhrzeit auf dem Server sehen. Zu der erhaltenen Uhrzeit müssen Sie zwei Stunden hinzurechnen, um die Uhrzeit mit der in unserer Zeitzone gleichsetzen zu können. Beim Erstellen der Screenshots war es also einmal 13:25:25 Uhr und einmal 13:35:12 Uhr.

Wichtiger Hinweis

Ein Problem stellt unter Umständen die verschlüsselte Verbindung mittels SSL dar. Bei SSL erfolgen Authentifizierungen über ein Zertifikat. Sollte PHP respektive cURL während des Versuchs, eine Anfrage zu stellen, abbrechen und Ihnen die Meldung

```
SSL certificate problem, verify that the CA cert is OK
```

ausgeben, kann das erhaltene Zertifikat nicht auf Gültigkeit überprüft werden. Sie benötigen dann eine Datei namens **ca-bundle.crt**, die Sie auf der dem Buch beiliegenden CD-ROM im Verzeichnis **x:\listings\xml\ebay** finden.

Um cURL mitzuteilen, wo es diese Datei finden kann, müssen Sie folgende Zeile auskommentieren:

```
// curl_setopt($chApi,CURLOPT_CAINFO,
     <vollständiger-pfad-zum-bundle>\ca-bundle.crt');
```

Wichtig ist, dass Sie auf jeden Fall einen vollständigen Pfad zum Bundle eintragen. Liegt die Datei **ca-bundle.crt** also im **htdocs**-Verzeichnis Ihres Webservers, könnte der Pfad folgendermaßen aussehen:

c:\apache group\apache2\htdocs\ca-bundle.crt

Die Angabe eines relativen Pfades führt dazu, dass cURL das Bundle *nicht* finden kann.

60.4 Zusammenfassung

- Webservices dienen zur Kommunikation zwischen zwei Maschinen.
- Es gibt zwei gängige Standards: XML-RPC und SOAP.
- Sowohl XML-RPC als auch SOAP nutzen als Protokoll HTTP, jedoch wird der Header des Protokolls in der Regel modifiziert.
- Beide Standards nutzen bei der Formulierung der Anfragen XML.
- Die eBay-API ist zwar ein Webservice, basiert aber auf keinem der beiden gängigen Standards. Anlehnungen sind jedoch zu finden.

60.5 Fragen und Übungen

1. Was ist WSDL?
2. Worin besteht der Unterschied zwischen einem auf XML-RPC basierenden Webservice und einem, der SOAP nutzt?
3. Wozu dient bei der eBay-API das Element `DetailLevel` im Header und dem Dokument?
4. Welche Bibliothek benötigen Sie in PHP, um Webservices wie z. B. die eBay-API nutzen zu können?
5. Was ist die »Sandbox«?

Index

B

C

D

F

G

H

P

R

S