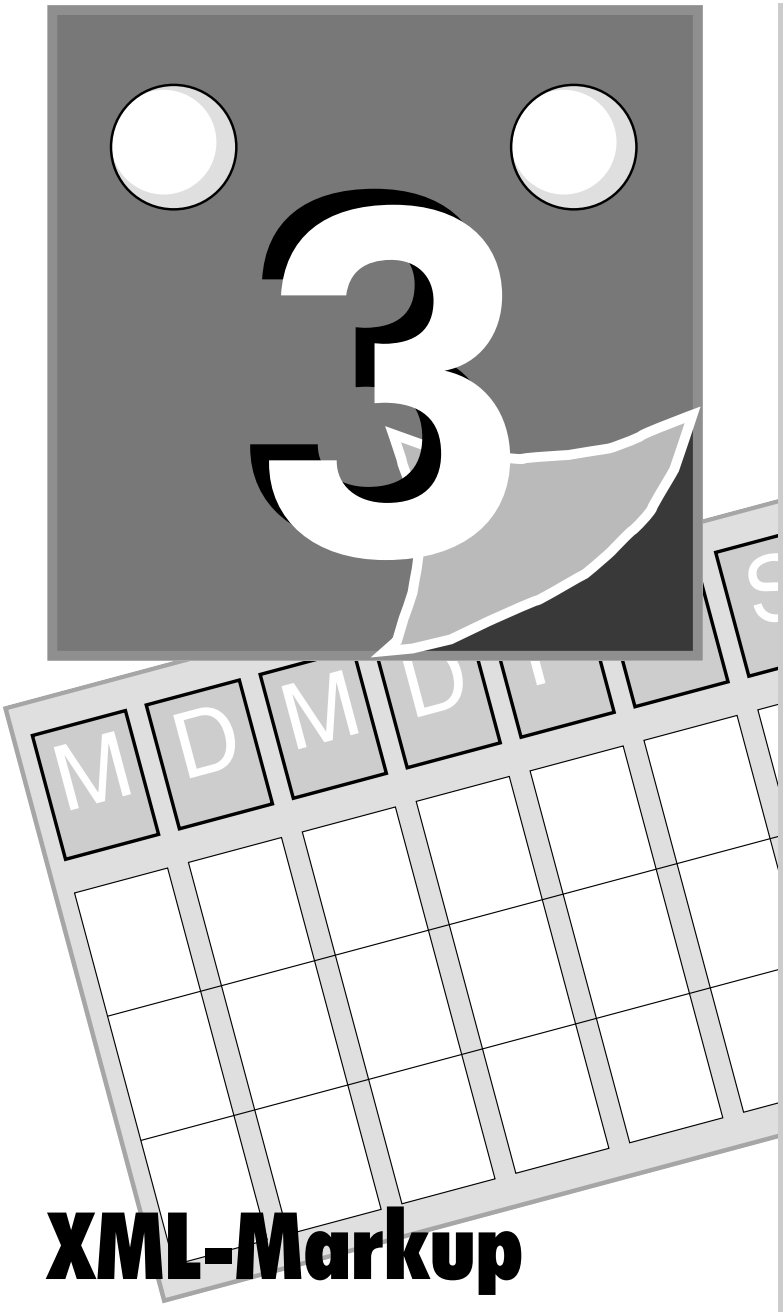

XML-Markup

Woche 1

In Kapitel 2 haben Sie die wichtigsten Funktionen des XML-Markups für Elemente und Entities kennengelernt. Dieses Kapitel baut darauf auf, und Sie lernen die folgenden Dinge kennen:

- Attribute
- Entity-Verweise und ihre Verwendung als Shortcuts
- Kommentare im Code
- CDATA-Abschnitte und ihre Verwendung
- Verarbeitungsanweisungen (Processing Instructions)

Markup-Begrenzungszeichen

Gestern haben Sie die Markup-Zeichen von XML ganz allgemein kennengelernt. Jetzt wollen wir ein bißchen technischer werden und die genauen Details der Markup-Deklarationen von XML betrachten.

Tabelle 3.1 zeigt die Bestandteile der Element-Tags von XML. Wenn die Details noch technischer werden, ist es wichtig, daß Sie mit diesen Dingen vertraut sind. (Sie müssen sie aber nicht unbedingt im Gedächtnis festschreiben!)

Symbol	**Beschreibung**
<	Öffnendes Begrenzungszeichen für das Start-Tag
</	Öffnendes Begrenzungszeichen für das Ende-Tag
foo	Elementname
>	Schließendes Begrenzungszeichen für ein Tag
/>	Schließendes Begrenzungszeichen für ein leeres Tag

Tabelle 3.1: Die Teile eines Element-Tags

Sie sollten sich merken, daß HTML einfach auf dem Erkennen vorprogrammierter Tags basiert, während bei XML durch diese Komponenten eine Verarbeitung der Element-Tags ausgelöst wird. Das Verhalten des XML-Prozessors und was er als nächstes erwartet, wird direkt durch die hier gezeigten Symbole gesteuert.

Element-Markup

XML hat mit Element-Markup zu tun. Das scheint eine offensichtliche Feststellung zu sein, aber sie soll hier wiederholt werden, weil sie auf den großen Unterschied zwischen XML als Markup-Sprache und einer beliebigen Tag-Sprache hinweist. Wie Sie bereits gesehen haben, tendiert HTML oft dazu, eine Tag-Sprache statt eine Markup-Sprache zu sein. Das ist eine direkte Konsequenz der Tatsache, daß die Web-Browser so bewußt locker mit der Akzeptanz fehlerhaften Markups umgehen.

Statt die XML-Tags als Kennzeichner zu betrachten, die anzeigen, wann ein Stil geändert werden soll oder eine neue Zeile beginnt, setzt sich das Element-Markup von XML aus drei Teilen zusammen: einem Start-Tag, dem Inhalt und dem Ende-Tag. Das sehen Sie in Tabelle 3.2. Das Start-Tag und das Ende-Tag können als eine Hülle verstanden werden, und wenn Sie sich ein Element vorstellen, dann sollten Sie ein mentales Bild eines Textabschnitts mit beiden Tags vor Augen haben.

Symbol	Name	Beschreibung
`<foo>`	Start-Tag	Am Anfang eines Elements, das öffnende Tag
`text`	Inhalt	In der Mitte eines Elements, sein Inhalt
`</foo>`	Ende-Tag	Am Ende eines Elements, das schließende Tag

Tabelle 3.2: Bestandteile eines Elements

Beachten Sie, daß der Elementname im Start-Tag genau dem Namen im Ende-Tag entsprechen muß. Beispielsweise wäre folgendes falsch:

```
<simple.element>Dieses Element wird nicht mehr geschlossen!</simple.Element>
```

In den ersten XML-Versionen vor der vollständigen Spezifikation wurde die Groß-/Kleinschreibung nicht berücksichtigt. Es gibt immer noch XML-Softwarepakete, die die Groß-/Kleinschreibung nicht beachten und die keinen Fehler monieren, wenn Sie Groß- und Kleinbuchstaben kombinieren. Der Konformität mit den XML-Anforderungen halber sollten Sie sorgfältig darauf achten, daß Sie Groß- und Kleinbuchstaben konsistent verwenden.

Attribut-Markup

Wie Sie gestern erfahren haben, werden Attribute genutzt, um Informationen an die in einem Element enthaltenen Informationen anzufügen. Die allgemeine Form der Verwendung eines Attributs sieht wie folgt aus:

```
<element.name eigenschaft="wert">
```

oder

```
<element.name eigenschaft='wert'>
```

Die technische Beschreibung des Markups in dieser Attributspezifikation finden Sie in Tabelle 3.3.

Symbol	Beschreibung
<	Öffnendes Begrenzungszeichen für das Start-Tag
element.name	Elementname
eigenschaft	Attributname
=	Wertindikator
"	Begrenzungszeichen für die Zeichenkette
'	Alternatives Begrenzungszeichen für die Zeichenkette
wert	Attributwert
<	Schließendes Begrenzungszeichen für das Start-Tag

Tabelle 3.3: Bereitstellung eines Attributs

Beachten Sie, daß ein Attributwert in Anführungszeichen eingeschlossen werden muß. Sie verwenden dazu entweder einfache Anführungszeichen (`<lie size='big'>`) oder doppelte Anführungszeichen (`<lie size ="massive">`), aber Sie können die beiden nicht innerhalb einer einzigen Spezifikation kombinieren.

Falls Sie ohne eine DTD arbeiten (keiner der in diesem Kapitel gezeigten XML-Codes bedingt, daß Sie dem XML-Dokument eine DTD zuordnen), können Sie das Attribut und seinen Wert einfach dann angeben, wenn Sie das Element zum ersten Mal verwenden, wie in Listing 3.1 gezeigt. Wenn Sie mehrfach Attribute für dasselbe Element angeben (wie in den Zeilen 3 und 4 von Listing 3.1), werden diese Spezifikationen einfach.

Listing 3.1: Attributspezifikationen

```
1: <?xml version="1.0"?>
2: <home.page>
3:  <para number="first">Dies ist der erste Absatz.</para>
4:  <para number='second' color="red">Dies ist
5:   der zweite Absatz.</para>
6:  </home.page>
```

Wenn der XML-Prozessor auf die Zeile 3 trifft, erkennt er, daß ein `para`-Element ein Zahlenattribut hat. (Sie wissen, daß dies alles in Abwesenheit einer DTD passiert, die explizit deklarieren würde, welche Attribute ein `para`-Element hat.) Wenn er dann auf Zeile 4 trifft, erkennt er, daß ein `para`-Element auch ein `color`-Attribut hat.

Es gibt ein Attribut, das für XML selbst reserviert ist: `xml:lang`. Dieses Attribut gibt an, in welcher natürlichen Sprache das Element geschrieben wurde. Der Wert des Attributs ist einer der Sprachencodes, die in ISO 639 festgelegt sind; einige der gebräuchlichsten sind in Tabelle 3.4 aufgelistet.

Code	**Sprache**
`ar`	Arabisch
`ch`	Chinesisch
`de`	Deutsch
`en`	Englisch
`es`	Spanisch
`fr`	Französisch
`gr`	Griechisch
`it`	Italienisch
`ja`	Japanisch
`nl`	Niederländisch
`pt`	Portugiesisch
`ru`	Russisch

Tabelle 3.4: Gebräuchliche ISO-639-Sprachencodes

Wenn es mehrere Versionen einer Sprache gibt, wie beispielsweise britisches und amerikanisches Englisch, kann dem Sprachencode ein Trennstrich (-) und einer der in ISO 3166 festgelegten Ländercodes folgen. Einige der gebräuchlichsten Ländercodes sind in Tabelle 3.5 aufgeführt. Wenn Sie bereits Erfahrung mit dem Internet haben, werden Sie einige dieser Codes aus E-Mail-Adressen und URLs bereits kennen. Ein Element, das in amerikanischem Englisch geschrieben wurde, könnte wie folgt gekennzeichnet werden (beachten Sie die Groß-/Kleinschreibung; der Sprachcode wird in Kleinbuchstaben, der Ländercode in Großbuchstaben angegeben):

```
<para xml:lang="en-US">Mein Land sieht so aus.</para>
```

Code	Land
AT	Österreich
BE	Belgien
CA	Kanada
CN	China
DE	Deutschland
DK	Dänemark
EN	England
ES	Spanien
FR	Frankreich
GR	Griechenland
IT	Italien
JA	Japan
NL	Niederlande
PT	Portugal
RU	Rußland
US	Vereinigte Staaten von Amerika

Tabelle 3.5: Gebräuchliche ISO 3166-Ländercodes

Die in den Tabellen 3.4 und 3.5 gezeigten Codes sind keineswegs vollständig. Es gibt noch ein weiteres Codierungsschema von der IANA (*Internet Assigned Numbers Authority*), das in RFC 1766 definiert ist. Und falls nötig, können Sie auch Ihren eigenen Sprachcode definieren. Benutzerdefinierten Codes muß der String `x-` vorausgehen, so daß Sie ein Element auch als »Computer-Chinesisch« ankündigen können, etwa wie folgt:

```
<para xml:lang="x-cc">Können Sie das entziffern?</para>
```

Namensregeln

Bisher haben Sie das Markup für Elemente und Attribute kennengelernt, und in allen Beschreibungen wird gesagt, daß diese Markup-Objekte Namen tragen. XML hat bestimmte Namensregeln für seine Markup-Objekte.

Die Namensregeln von XML sehen wie folgt aus:

- Ein Name besteht aus mindestens einem Zeichen: a bis z oder A bis Z.
- Falls der Name aus mehr als einem Zeichen besteht, kann er mit einem Unterstrich (_) oder einem Doppelpunkt (:) beginnen.

- Dem ersten Buchstaben (oder Unterstrich) können ein oder mehr Buchstaben, Ziffern, Trennstriche, Unterstriche, Punkte, kombinierende Zeichen, Erweiterungszeichen oder ignorierbare Zeichen folgen. (Diese letzten drei Zeichenklassen wurden aus dem Unicode-Zeichensatz übernommen und beinhalten einige spezielle Unicode-Zeichensymbole und Akzente. Eine vollständige Liste finden Sie online in der XML-Empfehlung unter `http://www.w3.org/XML/REC-xml`.)

Das W3C (*World Wide Web Consortium*) aktualisiert seine Web-Site in regelmäßigen Abständen, und die URLs für Empfehlungen, Hinweise und Arbeitsskizzen ändern sich relativ häufig. Wenn Sie ihre Web-Site besuchen, finden Sie einen Zeiger auf den URL-Minder-Service. Dieser kostenlose Dienst ist eines der vielen Wunder im Web. Durch Registrierung einer Webseite – einer beliebigen Webseite – erhalten Sie automatisch eine E-Mail, wenn sich irgend etwas auf dieser Seite ändert. Das ist eine ausgezeichnete Möglichkeit, sich über Weiterentwicklungen zu informieren.

Beachten Sie, daß in Elementnamen keine Leerzeichen und Tabulatoren erlaubt sind (`<eins zwei>` würde als zwei separate Namen interpretiert), und die einzigen erlaubten Interpunktionszeichen sind der Gedankenstrich (-) und der Punkt (.).

Es gibt keine Regel, die besagt, Ihre Namensgebung müsse sinnvoll sein. So lange Sie die Namensregeln nicht verletzen, können Sie den XML-Objekten beliebige Namen zuweisen, die so lang und so sinnlos sein dürfen, wie Sie das für richtig halten. Einer der wichtigsten Vorteile bei der Verwendung von XML ist, daß der Code selbstbeschreibend ist. Wenn Sie Elemente wie `<dingsda>`, `<irgendwas>` oder `<huh>` verwenden, dann machen Sie den ganzen Effekt zunichte. Versuchen Sie Namen zu verwenden, die wenigstens ansatzweise die Art oder die Aufgabe des Objekts treffen. Vergessen Sie nicht, daß eines der Ziele von XML die Lesbarkeit durch den Benutzer ist. Lesbar sein ist das eine, aber es ist durchaus sinnvoll, wenn die Namen auch eine Bedeutung kenntlich machen.

Kommentare

Keine Sprache, egal ob eine Programmiersprache oder eine Markup-Sprache, könnte überleben, würde sie nicht erlauben, dem Code Kommentare hinzuzufügen. Aus der Wartungsperspektive ist es sehr wichtig, dauerhafte Aufzeichnungen zu besitzen, die sagen, warum bestimmte Dinge auf ganz bestimmte Weise gemacht wurden. Die beste Methode, Ihren Code zu dokumentieren, ist die Aufnahme von Erklärungen in den Code mit Hilfe von Kommentaren.

In Übereinstimmung mit dem Entwurfsziel, XML einfach zu halten, sind auch die Kommentarfunktionen einfach. Kommentare haben die folgende Form:

```
<!-- Dies ist ein Kommentar -->
```

Das Start-Tag (<!--) und das Ende-Tag (-->) müssen genau wie hier gezeigt verwendet werden. Das Einfügen von Leerzeichen oder anderen Zeichen in diese Zeichenketten kann dazu führen, daß Tags oder alles andere innerhalb des Kommentars vom XML-Prozessor versehentlich als Markup interpretiert wird.

Vorausgesetzt, Sie verwenden das Start-Tag und das Ende-Tag für den Kommentar korrekt, wird alles innerhalb des Kommentartexts vom XML-Prozessor vollständig ignoriert. Der folgende Kommentar ist also ganz sicher:

```
<!-- Dies sind die Deklarationen für den <title> und den <body> -->
```

Es gibt nur eine Einschränkung, was Sie in Ihrem Kommentartext schreiben dürfen: Die Zeichenkette -- ist nicht erlaubt. Damit bleibt XML abwärtskompatibel zu SGML. (Die Zeichenkette --> beendet den Kommentar offensichtlich.)

Kommentare können an beliebiger Stelle in einem XML-Dokument angelegt werden – außerhalb anderer Markups. Das folgende ist also erlaubt:

```
<para>Das ist einfach <!-- sagen sie jedenfalls alle --> zu erledigen.</para>
```

während das folgende nicht geht:

```
<para <!-- unverschämte Lüge --> >Das ist ganz einfach.</para>
```

Zeichenverweise

Anders als SGML (und damit auch anders als HTML), das weitgehend auf ASCII basiert, wurde XML von Anfang an darauf hinentwickelt, auch andere Sprachen als Englisch zu unterstützen. XML unterstützt deshalb Zeichen mit Akzenten oder aus Fremdsprachen besser als SGML oder HTML.

In HTML können Sie immer den Code für das gewünschte Fremdsprachenzeichen eingeben (`è` wäre `&egrave;`, `í` wäre `&iacute;`, und `û` wäre `&ucirc;`). Wie Sie später in diesem Kapitel noch sehen werden, sind diese Codes letztlich Verweise auf Entities. Die Abkürzungen `egrave`, `iacute` und `ucirc` stammen aus dem Zeichensatz ISO 8859/1 (das ist der SGML-Zeichensatz), der sich von der Version ISO/IEC 646 des ASCII-Alphabets (den ersten 128 Zeichen) ableitet. ISO 8859/1 ist außerdem die Grundlage für die Schriften in Microsoft Windows.

Mit Hilfe dieser Verweise auf Zeichen-Entities können Sie zwar den meisten europäischen und skandinavischen Sprachen gerecht werden, aber Sie hätten plötzlich ein Problem, wenn Sie asiatische Sprachen oder Sprachen aus Mittelost darstellen wollten, wie etwa Japanisch, Hindi oder Arabisch. XML basiert jedoch auf Unicode und den noch umfangreicheren ISO/IEC 10646-Standards (die sogar die Verwendung chinesischer Zeichen erlauben). Sie brauchen sich jetzt nicht zu detailliert mit diesen Zeichensätzen auseinandersetzen (oder überhaupt nicht, wenn Sie vorhaben, im Web nur in westlichen Sprachen zu veröffentlichen), aber wir werden später noch einmal darauf zurückkommen.

Das Wichtigste, was Sie über diese exotischen Zeichen wissen sollten, ist, daß Sie sie auch eingeben können, wenn sie von Ihrer Tastatur nicht unterstützt werden. Dazu geben Sie einen Zeichenverweis ein.

Ein Zeichenverweis besteht aus der Zeichenkette `&#`, gefolgt von der Nummer des Zeichens im ISO/IEC 10646-Alphabet und abgeschlossen durch ein Semikolon (;). Die Nummer des Zeichens kann dezimal dargestellt werden, dann geben Sie sie einfach unverändert ein, oder hexadezimal, dann schreiben Sie den Buchstaben `x` vor die Zahl, beispielsweise `x12ABC`. Der Zeichenverweis für das Copyright-Symbol (©) beispielsweise – das in HTML als `&copy;` angesprochen wird –, ist `&#169;` (dezimal) oder `&#xA9;` (hexadezimal).

Vordefinierte Entities

Zeichenverweise erlauben Ihnen auch, Zeichen einzugeben, die auf Ihrer Tastatur nicht zur Verfügung stehen. Unter anderem geht es dabei um die *vordefinierten Entities*. Dabei handelt es sich um Zeichen, die Sie normal eingeben, was aber nicht empfehlenswert ist, weil sie sonst möglicherweise als Markup-Zeichen fehlinterpretiert werden. Um Ihr Gedächtnis aufzufrischen, sehen Sie in Tabelle 3.6 die vordefinierten Entities.

Zeichen	**Ersatz**
&	& oder &#38
'	' oder '
>	> oder >
<	< oder <
"	" oder "

Tabelle 3.6: Vordefinierte Entities

Sie können ein benanntes Entity eingeben, um ein Zeichen darzustellen, wie etwa `&apos;`, oder einen Zeichenverweis eintragen, wie etwa `&#39;`. Die Zeichenverweise für das Ampersand- (&) und das Kleiner-Zeichen (<) sind jedoch Sonderfälle; deshalb muß ein doppeltes Escape dafür erfolgen. Der Grund dafür wird im nächsten Abschnitt erklärt.

Entity-Verweise

Wie Sie aus der gestrigen Beschreibung des Aufbaus eines XML-Dokuments wissen, sind Entities normalerweise externe Objekte, wie etwa Grafikdateien, die in das Dokument eingebunden werden sollen. Um auf diese externen Entities zu verweisen, brauchen Sie eine DTD für Ihr XML-Dokument. Sie werden mehr über diese Entities erfahren, wenn Sie die DTDs kennenlernen, aber es gibt noch einen Entity-Typ, den Sie bereits benutzen, das sogenannte *interne Entity*. Damit können Sie sich eine Menge unnötiger Schreibarbeit ersparen.

Interne Entities sehen ganz ähnlich wie Zeichenverweise aus, mit einem entscheidenden Unterschied – Sie müssen ein internes Entity deklarieren, bevor Sie es nutzen können.

Entity-Deklarationen

Die Deklaration eines internen Entities hat die folgende Form:

```
<!ENTITY name "ersatztext">
```

Immer wenn die Zeichenkette `&name;` dann in Ihrem XML-Code erscheint, ersetzt der XML-Prozessor sie automatisch durch den Ersatztext (der beliebig lang sein darf). Sinnvoll eingesetzt, können Ihnen die Entity-Verweise eine Menge Schreibarbeit ersparen.

Die Vorteile von Entities

Sie können sich einen Entity-Verweis ähnlich einem Makro vorstellen. Aber wie auch immer Sie ihn bezeichnen, er kann Ihnen viel Zeit sparen, wenn es einen Textabschnitt gibt, den Sie mehrere Male brauchen, oder wenn Sie irgendeine Art Schablonentext verwenden wollen.

Betrachten Sie das Beispiel in Listing 3.2. Dort wird ein Entity-Verweis für einen Copyright-Hinweis verwendet.

Listing 3.2: Verwendung eines internen Entity

```
1:  <?xml version="1.0"?>
2:  <home.page>
3:    <head><title>Title Page</title></head>
4:    <body> <h1>The Title Page</h1>
5:      <para>(c) 1998, &rights;</para>
6:     </body>
7:  </home.page>
```

Mit der folgenden Deklaration für das Entity `rights`:

```
<!ENTITY rights "Alle Rechte vorbehalten. Kein Teil dieses Buchs, auch nicht der
Entwurf, das Cover-Design und die Icons, dürfen ohne vorherige Zustimmung des
Verlags in irgendeiner Form über irgendein Medium (elektronisch, durch Kopien,
Aufzeichnungen oder andere Methoden) reproduziert oder weitergegeben werden."
```

Damit ergibt sich durch Ersetzung in Zeile 5 von Listing 3.2 das folgende Ergebnis:


```
<para>(c) 1998, Alle Rechte vorbehalten. Kein Teil dieses Buchs, auch nicht der
Entwurf, das Cover-Design und die Icons, dürfen ohne vorherige Zustimmung des
Verlags in irgendeiner Form über irgendein Medium (elektronisch, durch Kopien,
Aufzeichnungen oder andere Methoden) reproduziert oder weitergegeben werden.</
para>"
```

Bei dieser Verwendung des Entity-Verweises müssen Sie den Text nur ein einziges Mal eingeben, nämlich in der Entity-Deklaration, und brauchen dann bei einer Bearbeitung nicht mehr jedes Vorkommen der Zeichenkette im Text zu suchen und zu ändern. Auf diese Weise vereinfachen die Entity-Verweise die Aufgabe, XML-Dokumente zu erzeugen und zu verwalten. In Kapitel 8 erfahren Sie, wie Sie diese Funktion so erweitern können, daß externe Entities als Ausgangspunkt für Textfunktionen genutzt werden, und wie Sie diese Text-Entities in einem allgemeinen Dokument deklarieren, auf das beliebig viele andere Dokumente Zugriff haben.

Gefahren bei der Verwendung von Entities

Sie haben jetzt gesehen, wie praktisch interne Entity-Verweise als Abkürzung für die Eingabe von Textabschnitten und die Arbeit mit variablem Inhalt sein können. Offensichtlich können Ihnen Entity-Verweise bei der richtigen Handhabung und den entsprechenden Vorbereitungen viel Zeit und Mühe ersparen.

Natürlich stellt man bei so einer praktischen Funktion sofort eine Frage: »Könnte ich sie auch nutzen, um Markups damit einzufügen?« Das ist eine gute und naheliegende Idee. Können Sie Markups innerhalb des Ersatztexts angeben? Ja, das können Sie ..., aber es unterliegt einigen Einschränkungen, und Sie müssen sorgfältig darüber nachdenken, um unliebsame Überraschungen zu vermeiden.

Als erstes müssen Sie daran denken, daß XML den Inhalt des Entity-Ersatztexts verarbeitet, wenn es den Entity-Verweis expandiert. Das bedeutet, Sie dürfen im Ersatztext keine Escapes für Markup-Zeichen verwenden; Sie müssen für diese Zeichen ein doppeltes Escape angeben. Betrachten Sie ein einfaches Beispiel:

```
<!ENTITY gefaehrlich "Black &#38; White">
```

Wenn der XML-Prozessor den Entity-Verweis `&gefaehrlich;` im XML-Dokument sieht, expandiert er sofort das vordefinierte Entity, bevor er den Ersatztext einfügt. Dieser XML-Code scheint ganz harmlos zu sein:

```
<text>Dieser Text ist nicht &gefaehrlich;.</text>
```

Aber sehen wir uns schrittweise an, was passiert:

1. Der XML-Prozessor sieht den Verweis `&gefaehrlich;` und sucht nach dem Ersatztext.
2. Er findet `Black &#38; White` und macht daraus `Black & White`.
3. Der XML-Prozessor fügt den Ersatztext ein, und der resultierende XML-Code sieht wie folgt aus:

   ```
   <text>Dieser Text ist nicht Black & White.</text>
   ```

4. Der XML-Prozessor versucht dann, das Ampersand zu verarbeiten, und gibt einen Fehler zurück, weil `&` nicht als Entity deklariert wurde.

Die Fallstricke und wie man sie vermeidet

Sie haben einige der Probleme kennengelernt, die entstehen, wenn der Inhalt von Entity-Verweisen dereferenziert wird. Im schlimmsten Fall können sie Ihren gesamten XML-Code zerstören. Es gibt natürlich eine Möglichkeit, diese Probleme zu vermeiden – durch ein doppeltes Escape für jedes Markup im Ersatztext, beispielsweise wie folgt:

```
<!ENTITY sicher "Harry &#38;#38; Fred &amp;amp; Joe">
```

Wenn der XML-Prozessor den Entity-Verweis `&sicher;` in diesem XML-Dokument sieht:

```
<text>Der Job konnte &sicher; übertragen werden.</text>
```

ergibt die Erweiterung korrekten Code. Betrachten wir, was passiert, wenn der XML-Prozessor den Entity-Verweis dereferenziert:

1. Der XML-Prozessor sieht den Entity-Verweis `&sicher;` und sucht nach dem Ersatztext.
2. Der XML-Prozessor findet »Harry & Fred & Joe« und dereferenziert es zu Harry & Fred & Joe.
3. Der XML-Prozessor fügt den Ersatztext ein, und es entsteht der XML-Code

   ```
   <text>Der Job konnte Harry &amp;#38; Fred &amp;amp; Joe übertragen werden.</text>
   ```

4. Der XML-Prozessor parst den resultierenden Code, sieht die Entity-Verweise und dereferenziert sie wie folgt:

```
<text>Der Job konnte Harry & Fred & Joe übertragen werden.</text>
```

Wie Sie aus diesem Beispiel erkennen, können Sie das Escape für das Markup mit Hilfe des Entity-Verweises (im Beispiel `&#38;`) oder als Zeichenverweis (`&amp;`) auf das vordefinierte Entity darstellen.

Synchrone Strukturen

Neben diesen Problemen gibt es noch eine sehr wichtige Einschränkung bei der Verwendung von Markup in Entities. In Kapitel 2 haben Sie erfahren, daß die logische und die physische Struktur in einem XML-Dokument synchron sein müssen.

Damals konnten Sie mit dieser Einschränkung noch nicht sehr viel anfangen, weil man sich kaum vorstellen kann, daß die beiden Strukturen *nicht* synchron sind. Deshalb zeigen wir Ihnen hier ein Beispiel, bei dem die beiden Strukturen asynchron werden: Die logische Struktur setzt sich aus den Elementen im XML-Dokument und im Ersatztext zusammen. Die physische Struktur setzt sich aus dem Dokument-Entity (dem Wurzel-Entity des XML-Dokuments mit dem Entity-Verweis) und dem internen Entity (dem Ersatztext) zusammen. Die beiden Objekt sind diskrete physische Entities, was XML betrifft, obwohl sie wie in diesem Fall tatsächlich innerhalb einer einzigen Datei abgelegt sind. Damit die beiden Strukturen synchron bleiben, muß jedes Element des Ersatztextes dort beginnen und auch dort enden (mit anderen Worten: innerhalb des Entities).

Das folgende wäre also erlaubt:

```
<!ENTITY sicher "&amp;#38;#60;emph&amp;#62;Harry&amp;#38;#60;/emph&amp;#62; und Joe">
<text>Der Job konnte &sicher; übertragen werden.</text>.
```

weil der dereferenzierte Entity-Verweis folgendes ergäbe:

```
<text>Der Job konnte <emph>Harry</emph> und Joe übertragen werden.</text>
```

Das folgende dagegen könnte zu vielen Problemen führen:

```
<!ENTITY unsicher ""&#38#60;emph&#62;Harry und Joe">
<text>Der Job konnte &unsicher;</emph> übertragen werden.</text>
```

Auch wenn der Entity-Verweis dereferenziert würde, wäre das resultierende Markup noch irgendwie gültig:

```
<text>Der Job wurde <emph>Harry und Joe</emph>übertragen.</text>
```

Obwohl wir immer noch über *interne* Entities sprechen, die völlig unserer Kontrolle unterliegen, sind die Einschränkungen relativ logisch. Derselbe Dereferenzierungsmechanismus wird für externe Entities angewendet, und wenn man das Entwurfsziel der einfachen Nutzung im Web von XML berücksichtigt, haben wir absolut keine Kontrolle darüber, was in externen Entities enthalten ist. Die XML-Entwickler hätten einen Unterschied zwischen internen und externen Entities einführen können, aber das würde gegen zwei der wichtigsten Entwurfsziele von XML verstoßen – Einfachheit und Klarheit.

Wo Entities deklariert werden

Sie haben jetzt erfahren, wie ein interner Entity-Verweis aussieht, und Sie haben einige der Vorteile und Nachteile bei der Verwendung von Entity-Verweisen gesehen. Bevor wir weitergehen, müssen Sie noch lernen, wo die Entity-Deklarationen erfolgen.

Entity-Verweise sind normalerweise nur in der DTD des XML-Dokuments erlaubt. Die Deklarationen von Elementstrukturen und Entities sind der einzige Grund dafür, warum überhaupt eine DTD verwendet wird. Später erhalten Sie detailliertere Informationen über DTDs; alles, was Sie hier wissen müssen, finden Sie in Listing 3.3 erklärt.

Listing 3.3: Deklaration eines internen Entity

```
1:  <?xml version="1.0"?>
2:  <!DOCTYPE home.page [
3:    <!ENTITY shortcut "Dies ist der Ersatz.">
4:  ]>
5:  <home.page>
```

Zeile 1 von Listing 3.3 enthält die jetzt schon bekannte XML-Deklaration. Zeile 2 ist eine Dokumenttypdeklaration. Diese Zeile wird später für die Zuordnung zwischen dem XML-Dokument und der DTD, die seine Struktur beschreibt, herangezogen.

Die Dokumenttypdeklaration ist die XML-Anweisung, die deklariert, welche Art XML-Dokument folgt, und die die DTD (Dokumenttypdefinition) angibt, die die Beschreibung der für diesen Dokumenttyp erlaubten Struktur enthält. (Diese beiden Begriffe werden leicht verwechselt.)

Die Dokumenttypdeklaration hat die folgende Form:

```
<!DOCTYPE name externer.zeiger [ interne.untermenge ]>
```

Dabei zeigt `externer.zeiger` auf eine separate Datei, die die externe Untermenge der DTD enthält. Machen Sie sich hier noch keine Gedanken darüber; der Trick dabei ist, daß Sie den Zeiger auch weglassen und sich auf die interne Untermenge der DTD konzentrieren können. In diesem Fall brauchen Sie die folgende Deklaration:

```
<!DOCTYPE name [ interne.untermenge ]>
```

In dieser internen Untermenge können Sie beliebige viele Elemente, Attribute und Entities deklarieren, ohne eine externe DTD zu benötigen.

Wie Sie später noch erfahren werden, gibt es zahlreiche weitere Tricks, die mit der internen DTD-Untermenge möglich sind. Alles, was Sie in der internen Untermenge ablegen, hat Priorität vor allen Inhalten externer Untermengen. Beispielsweise können Sie eine Standardmenge globaler Werte für eine ganze Folge von XML-Dokumenten deklarieren und dann die globalen Werte in jedem einzelnen XML-Dokument ganz individuell überschreiben, aber das ist eine andere Geschichte.

Bevor wir das Thema der DTDs verlassen, gibt es noch eines, was Sie sich angewöhnen sollten, auch wenn es hier scheinbar noch keinen Sinn ergibt. Sie verwenden zwar noch keine externe DTD, aber wenn Sie es tun, muß der Name, den Sie dem Dokumenttyp geben, derselbe wie der Name des Wurzel-Elements im XML-Dokument sein. Das sehen Sie in Listing 3.3, wo der Dokumenttypname (`home.page` in Zeile 2) derselbe wie der (erste) Wurzel-Elementname (Zeile 5) ist. Das ist nicht erforderlich, wenn es keine externe DTD gibt, aber Sie sollten sich diesen Stil dennoch angewöhnen.

CDATA-Abschnitte

Sie haben erfahren, wie man ein Escape für Markup-Zeichen anlegt, und zwar indem man die vordefinierten Entities und Zeichenverweise nutzt. Das Ersetzen jedes Markup-Zeichens in einem Textabschnitt könnte ein langwieriger und mühseliger Prozeß sein. Darüber hinaus könnte es Situationen geben, wo Sie möchten, daß diese Zeichen so bleiben, wie sie sind (etwa wenn Sie den XML-Code für die weitere Verarbeitung durch eine andere Applikation weitersenden).

In diesem Fall verwenden Sie einen CDATA-Abschnitt (Character Data), etwa wie folgt:

```
<![CDATA[Dies ist der Text < 5 Zeilen >, den der &!%# XML-Prozessor nicht
verändern soll!]]>
```

Nichts, absolut nichts, was zwischen dem öffnenden Tag (`<![CDATA[`) und dem schließenden Tag (`]]>`) steht, wird als Markup erkannt. Sie müssen kein Escape für Markup-Zeichen in einen CDATA-Abschnitt einfügen. (Sie können das nicht einmal, weil auch das Escape nicht erkannt würde.) Das einzige, was erkannt wird, ist das Tag für das Abschnittsende (/]]>); diese Zeichenkette darf also offensichtlich nicht in einem CDATA-Abschnitt erscheinen. Als logische Konsequenz erkennen Sie, daß CDATA-Abschnitte nicht ineinander verschachtelt werden dürfen.

Die Verwendung von Markup-Zeichen in einem solchen CDATA-Abschnitt in einem XML-Dokument, das um das Markup herum aufgebaut ist, geht einem eigentlich gegen den Strich. Ein XML-Prozessor soll Sie daran hindern, dieses ungeschriebene Gesetz zu brechen, und er verzeiht keine Fehler. Die öffnende und die schließende Zeichenkette eines CDATA-Abschnitts müssen genau so geschrieben werden wie hier gezeigt. Die kleinste Abweichung, ein Leerzeichen oder ein Tabulator irgendwo innerhalb der Zeichenkette, wird sofort bestraft. Der Inhalt des CDATA-Abschnitts wird dann entweder als Markup betrachtet, oder Ihr restliches Dokument (bis zum nächsten korrekt abgeschlossenen CDATA-Abschnitt) wird als Teil des CDATA-Abschnitts betrachtet, und das gesamte Markup wird ignoriert. Seien Sie also gewarnt!

CDATA-Abschnitte sind eine der empfohlenen Methoden, Applikationscode (JavaScript, VBasic-Code, Perl-Code usw.) in Ihren XML-Code einzubetten. Sie könnten den eingebetteten Code auch in Kommentare einschließen, aber es ist nicht garantiert, daß der XML-Prozessor einer Applikation auch den Kommentartext übergibt. Es besteht also das Risiko, daß der Kommentarinhalt entfernt wird, bevor die Applikation den Code sieht.

Es ist zwar durchaus erlaubt, Ihren eigenen Elementtyp zu deklarieren, der den eingebetteten Code aufnimmt (wie das <script>-Element in HTML 4), damit brechen Sie aber implizit den Geist des generischen Markups. Es wäre auch nicht hilfreicher, wenn Ihr eingebetteter Code Zeichen enthielte, die als Markup interpretiert werden könnten, weil der Inhalt dieser Elemente vom XML-Prozessor auf die normale Weise interpretiert würde.

Die andere, vielleicht sogar beste, Methode, Code einzubetten, ist die Verwendung von Verarbeitungsanweisungen.

Verarbeitungsanweisungen (Processing Instructions)

Sie haben sicher schon Verarbeitungsanweisungen gesehen – ohne sie als solche zu erkennen. Die XML-Deklaration am Anfang jedes XML-Dokuments (die sich dort zumindest befinden *sollte*) ist eine Verarbeitungsanweisung:

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

XML-Markup soll generisch sein, und üblicherweise ist es das auch. Es gibt jedoch immer Situationen, in denen Sie Anweisungen für bestimmte Applikationen eingeben müssen. Eine solche Applikation könnte etwa ein Skript-Interpreter sein, deshalb sind Verarbeitungsanweisungen ähnlich wie CDATA-Abschnitte gut dafür geeignet, Code einzubetten. Während CDATA-Abschnitte einzig eine Methode darstellen, die Interpretation von Zeichen als Markup zu vermeiden, sind Verarbeitungsanweisungen direkt auf Ihre Applikation zugeschnitten, was noch viel besser ist. Beispielsweise könnten Sie auf diese Weise zwei oder mehr Mengen eingebetteten Skriptcodes haben, die jeweils für unterschiedliche Prozessoren oder Interpreter vorgesehen sind, und sie separat kennzeichnen, wie in Listing 3.4 gezeigt.

Listing 3.4: Eingebetteter Code in Verarbeitungsanweisungen

```
1:  <para>Dieser Text enthält zwei
2:  Verarbeitungsanweisungen,
3:    <?javascript Kann ich hier schreiben, was ich will?>
4:    <?perl Und hier auch?>
5:  eine für jeden Interpreter.</para>
```

Es gibt keine Einschränkungen bezüglich des Inhalts von Verarbeitungsanweisungen (der XML-Prozessor betrachtet den Inhalt nicht einmal als Bestandteil der Zeichendaten des Dokuments), aber der dafür gewählte Name muß den Namensregeln von XML gehorchen.

Zusammenfassung

In diesem Kapitel haben Sie Details über die Markup-Sprache XML kennengelernt. Außerdem haben Sie erfahren, wie man interne Entities deklariert und nutzt und welche Vorteile und Gefahren sie bergen. Sie haben die Zeichenverweise kennengelernt, mit denen Zeichen eingegeben werden können, die sich nicht auf Ihrer Tastatur befinden, und Sie haben gesehen, wie Sie mit Hilfe von Zeichenverweisen und vordefinierten Entities in Ihren Zeichendaten die Zeichen nutzen können, die normalerweise für das Markup reserviert sind.

Schließlich haben Sie erfahren, wie man Kommentare und CDATA-Abschnitte nutzt, um Text zu verbergen, der durch den XML-Prozessor als Markup interpretiert werden könnte, und wie man das Ganze noch erweitert, indem man mit Hilfe von Verarbeitungsanweisungen Code weitergeben kann, der für die Verarbeitung durch andere Applikationen vorgesehen ist.

F&A

F Welche der folgenden Elementnamen sind gültig, welche nicht?

a) `<para 1>`

b) `<para,1>`

c) `<para.1>`

d) `<Pa3A1>`

e) `<para!>`

A *Nur c und d sind erlaubt; a enthält ein Leerzeichen, b enthält ein Komma, und e enthält ein Ausrufezeichen.*

F Was stimmt im folgenden Codeabschnitt nicht?

```
<para size="12pt">'twas brillig and
       the slithey toves <!-- I've no idea
        what these are --> did gyre and gymble
        in the wabe.</para>
```

A *Kommentare dürfen nicht innerhalb von Elementen angelegt werden. Sie müssen sich außerhalb von Markup befinden.*

F Wo werden Entities deklariert?

A *Sie können Entities innerhalb der internen Untermenge oder der externen Untermenge der DTD deklarieren. Wenn Sie eine externe DTD verwenden, müssen Sie eine vollständige DTD anlegen. Falls Sie nur die Entities und sonst nichts brauchen, können Sie mit der internen DTD-Untermenge auskommen. Entity-Verweise in XML-Dokumenten, die externe DTD-Untermengen haben, werden nur ersetzt, wenn das Dokument analysiert wird.*

F Warum brauche ich eine XML-Deklaration? Es ist doch offensichtlich, daß es sich um XML-Code handelt.

A *Genau gesagt, braucht man gar keine XML-Deklaration. XML wurde auch als MIME-Typ anerkannt, d.h. wenn Sie den korrekten MIME-Header (*`xml/text` *oder* `xml/application`*) einfügen, kann ein Web-Server die darauffolgenden Daten explizit als XML-Dokument erkennen, unabhängig davon, was in dem Dokument steht. (MIME, Multipurpose Internet Mail Extensions, ist ein Internet-Standard für die Übertragung von Daten beliebigen Typs über E-Mail. Er definiert, wie Nachrichten formatiert und aufgebaut werden, kann den Typ und die Art des Nachrichteninhalts anzeigen und Informationen des internationalen Zeichensatzes beibehalten. MIME-Typen werden von Web-Servern verwendet, um die Daten in einer Antwort auf eine Suchabfrage zu kennzeichnen.)*

Die XML-Deklaration ist aus praktischen Gründen nicht zwingend; SGML- und HTML-Code kann häufig ganz einfach in perfekten XML-Code umgewandelt werden (falls er es noch nicht ist). Wäre die XML-Deklaration zwingend, wäre das nicht möglich.

F Kann ich Entities in Attributwerten sowie im Inhalt verwenden? Damit könnte ich Elemente parametrisieren.

A *Ja und nein. Sie können Entity-Verweise als Attributwerte angeben, aber ein Entity kann kein Attributwert sein. Es gibt strenge Regeln darüber, wo Entities verwendet werden können, und wann sie erkannt werden. Manchmal werden sie nur erkannt, wenn das XML-Dokument ausgewertet wird. Weitere Informationen finden Sie in der XML-Empfehlung unter* `http://www.w3.org/XML/REC-xml`.

F Kann ich binäre Daten in einen CDATA-Abschnitt schreiben?

A *Technisch gesehen gibt es keinen Hinderungsgrund, obwohl es sich eigentlich um einen Zeichendatenabschnitt handelt. Weil der XML-Prozessor den Inhalt eines CDATA-Abschnitts nicht als Teil der Zeichendaten des Dokuments betrachtet, weiß er nicht, was Sie dort geschrieben haben, und es ist ihm auch egal. Sie müßten sonst auch mit größeren Dateien und allen möglichen Übertragungsproblemen leben. Letztlich wäre es unsinnig, die Portabilität Ihrer XML-Dokumente zu gefährden, wenn es eine XML-Funktion gibt, die Sie für diese Zwecke nutzen können. Entities, die Sie in Kapitel 8 genauer kennenler-*

nen, erlauben Ihnen, ein Format und eine Hilfsanwendung für die Verarbeitung einer Binärdatei (möglicherweise eine Anzeige) und ihre Zuordnung zu einem XML-Dokument per Verweis zu deklarieren.

Übungen

1. Der folgende Codeausschnitt enthält zwei Fehler. Erkennen Sie sie?

   ```
   <![CDATA [Dies ist das verborgene &Markup!] ]>
   ```

   Überprüfen Sie Ihre Antworten, indem Sie den Code in einem der XML-Parser ausführen, was in Kapitel 5 noch weiter erklärt wird.

2. Gestern haben Sie das Markup für eine E-Mail-Nachricht angelegt. Ändern Sie das Markup unter Verwendung geeigneter Entities so, daß der XML-Code zu einer Grundlage für beliebige E-Mail-Nachrichten wird.