

84 Chapter 2 • XML and Data Content

```
the <hobbytopic>model railroad</hobbytopic> is now
operational. The operational aspects are left to the
railroad modeler's imagination.</p>
</modelrailroad>
```

The modified example now has many more pairs of tags scattered throughout the XML document. This insertion of tags within character data is legitimate, still creating a well-formed XML document. All these tag pairs do is denote certain properties to the text itself, allowing the words to take on more meaning than just their dictionary definitions. In the case of indexing, it is important for the XML application to know which words from the text need to be stored off to a special collection for searching against and under which categories these words should accept hits.

The tag pair that is scattered throughout the XML document is `hobbytopic`, which the Webmaster deemed should be used whenever a word is to be indexed against the term *hobby*. The index collection will take note of where in the XML document the indexed item was located and from which XML document it came. The root tag itself has attributes to assist the indexing program. The attributes are `subject` and `subjectname`, which allow for a broad categorization of this XML document. Otherwise, the remainder of the XML document is similar to the original model railroader magazine sample.

This tutorial has shown that authoring XML documents requires the author to know the category of application for which it is being authored. The application will drive the structure of the document, whether bulk character data is sufficient or if the data needs to be reduced into finer elements with many attributes. Do not confuse this concept with using XML documents in multiple applications, as discussed at the start of this chapter, where a single XML document may take on various views depending on what a parser or application has been instructed to do.

PROLOG AND PROCESSING INSTRUCTIONS

One of the last types of tags used in authoring XML documents not yet discussed is the **processing instruction**. The XML Recommendation defines processing instructions as containing instructions for applications that need to be passed through to the application. This means that the processing instruction does not contain any character data that will be output as marked-up text. The tag will include the name of the application (the target) that the tag was directed toward, so only the interested application will take note of the instructions. One can think of processing instructions as comment tags for XML parsers instead of humans.

A common declaration that appears to be a processing instruction is the **XML document declaration**, using the format below.

```
<?xml version = "1.0" standalone = "no" encoding = "UTF-8"?>
```

The XML recommendation does not actually consider the XML document declaration to be a Processing Instruction, rather it is an entity unto itself. The recommendation also provides a synonym, that being the XML Text Declaration. Many XML documents will include the XML document declaration. When it appears, it *must* be the very first line, before the root tag. It declares that the document is an XML document and provides some parsing assis-

tance as to how the document should be parsed. This example XML document declaration also declares that the referenced document conforms to the W3C XML 1.0 recommendation, relies on all external definition documents be read in to assist in parsing, and uses eight-bit character encoding, commonly known as ASCII.

The XML document declaration is the start of a **prolog**. The XML document concept of the prolog is similar to that of HTML. The typical HTML document starts with a <head> tag to define the header of the document, and follows up with a <body> tag to declare that the remainder of the document contains character data and formatting information to guide the browser in displaying the Web page. The XML implementation does not provide for an explicit tag set to denote the prolog and document sections of the file. Rather, the XML declaration tag indicates the start of the prolog, whereas the root element indicates where the XML document starts. The main purpose of XML prolog data is to provide the source of information that will guide the parser in interpreting the document contents according to the author's design.

To use the XML declaration tag properly, one needs to know what the purpose of its attributes is. First of all, there is a version attribute, indicating which W3C XML version recommendation the document was written against. The version attribute must be included in every instance of an XML document declaration. At the time of this writing, the only acceptable values are "1.0" and "1.1". The version attribute allows the XML parser to recognize whether the author included any newer XML features that the earlier recommendations did not provide for. This means that the XML author cannot mix and match entities based upon different versions of the XML recommendation. An XML document can be considered well-formed and possibly valid even when the XML document declaration is not provided. The same follows for external entities, that they should provide an XML text declaration, but is not a specific requirement.

The second XML document declaration attribute is named "encoding". It is optional, but must appear immediately after the version attribute if provided. The purpose of the encoding declaration is to inform the parser which character set to deploy, based on standard notation the W3C XML 1.0 recommendation. The encoding attribute is not required if the XML document is made up of ASCII characters. If an encoding attribute is not provided, then the parser is to use UTF-8 encoding, in which the ASCII character set is but a subset. The other common encoding is UTF-16, which can be considered the Unicode character set. The encoding attribute is usually used when one needs to provide the XML document using characters for non-Latin languages, such as Katakana for Japanese documents.

The third XML document declaration attribute is named "standalone". This attribute is also optional, but must appear as the final name-value pair of the XML declaration tag if provided. Only validating XML parsers use this attribute, and its value is ignored if the XML document is not being validated. This attribute can inform the parser whether this document can be parsed without the aid of external XML grammars being read in first to parse against. This is not the indicator that there will be no DTD, XML grammars, schemas or vocabularies that need to be read in. If there are no XML grammars to be read in at all, then this attribute can be left out altogether. This attribute is meant more for increasing the performance of XML applications. Once an XML document has been successfully processed in an XML application, subsequent parsing of the exact XML document may not require reading of external documents with their subsequent validation, speeding up application throughput.

Occasions when the standalone attribute must be provided with a "yes" value for externally referenced declarations of the following:

- 1-Attributes are provided with default values.
- 2-References to external entities are present in the XML document content..
- 3-Attributes are provided that will evaluate to a new value if normalized..
- 4-Elements with child elements are defined using white space..

In other instances the XML parser will treat the standalone attribute as if it had a "no" value. If all grammatical information is provided for in the XML document's internal subset, then one will not need to worry about the standalone declaration.

Leaving the XML declaration element, the processing instruction is dependent on which application will parse the document. A common processing instruction is the style sheet instruction, such as the one shown:

```
<?xml:stylesheet type = "text/xsl" href = "modelrailroad.xsl"?)>
```

This tag is directed to the parser (which normally handles style sheets too) and instructs it to look for a style sheet document named modelrailroad.xsl, located in the same address from which the original XML document was read. The purpose of the style sheet document is to direct the formatting of the output from the XML parser according to the rules as laid out in the modelrailroad.xsl document. Style sheet applications will be covered in Chapters 5–7. For this specific attribute, the type attribute informs the parser that the document is in text format, and the href attribute declares the style sheet source address.

Processing instructions were deliberately designed to act as comments for applications to prevent the misuse of comments that is rampant in the HTML environment. HTML was extended frequently through the creation of nonstandard tags embedded within HTML documents as comments. Although comments are considered extraneous to HTML documents just as they are in XML documents, stripping comments from an HTML document could result in the document no longer having the same meaning as it did with the comments. This is due to the possibility of application-specific tags being removed under the guise of unessential comments. Because the application-specific comments are handled in XML as processing instructions, there is no longer a chance that the content of any XML document can be altered by stripping out the comments. The processing instructions will remain and function as before.

An application-specific processing instruction could inform the application that it needs to divide a series of numbers by 100 because they need to be converted to decimal dollars. Such an example is presented below.

```
<?quacken fudgefactor = "100"?>
<checkingaccount>
<deposit xactiondate = "02-08-02">100000</deposit>
<check xactiondate = "02-12-02" number = "151" to =
"Warehouse Mart">5478</check>
<check xactiondate = "02-23-02" number = "152" to =
"Doctor Smith">4500</check>
</checkingaccount>
```