File Formats

Overview

This chapter describes a number of modules that are used to parse different file formats.

Markup Languages

Python comes with extensive support for the *Extensible Markup Language* XML and *Hypertext Markup Language* (HTML) file formats. Python also provides basic support for *Standard Generalized Markup Language* (SGML).

All these formats share the same basic structure (this isn't so strange, since both HTML and XML are derived from SGML). Each document contains a mix of *start tags*, *end tags*, plain text (also called character data), and *entity references*.

```
<document name="sample.xml">
    <header>This is a header</header>
    <body>This is the body text.  The text can contain
    plain text (&quot;character data&quot;), tags, and
    entities.
    </body>
</document>
```

In the above example, **<document>**, **<header>**, and **<body>** are start tags. For each start tag, there's a corresponding end tag which looks similar, but has a slash before the tag name. The start tag can also contain one or more *attributes*, like the **name** attribute in this example.

Everything between a start tag and its matching end tag is called an *element*. In the above example, the **document** element contains two other elements, **header** and **body**.

Finally, **&quot;** is a character entity. It is used to represent reserved characters in the text sections (in this case, it's an ampersand (**&**) which is used to start the entity itself. Other common entities include **&lt;** for "less than" (<), and **&gt;** for "greater than" (>).

While XML, HTML, and SGML all share the same building blocks, there are important differences between them. In XML, all elements must have both start tags and end tags, and the tags must be properly nested (if they are, the document is said to be *well-formed*). In addition, XML is case-sensitive, so **<document>** and **<Document>** are two different element types.

HTML, in contrast, is much more flexible. The HTML parser can often fill in missing tags; for example, if you open a new paragraph in HTML using the **<P>** tag without closing the previous paragraph, the parser automatically adds a **</P>** end tag. HTML is also case-insensitive. On the other hand, XML allows you to define your own elements, while HTML uses a fixed element set, as defined by the HTML specifications.

SGML is even more flexible. In its full incarnation, you can use a custom *declaration* to define how to translate the source text into an element structure, and a *document type description* (DTD) to validate the structure, and fill in missing tags. Technically, both HTML and XML are *SGML applications*; they both have their own SGML declaration, and HTML also has a standard DTD.

Python comes with parsers for all markup flavors. While SGML is the most flexible of the formats, Python's **sgmllib** parser is actually pretty simple. It avoids most of the problems by only understanding enough of the SGML standard to be able to deal with HTML. It doesn't handle document type descriptions either; instead, you can customize the parser via subclassing.

The HTML support is built on top of the SGML parser. The **htmllib** parser delegates the actual rendering to a formatter object. The **formatter** module contains a couple of standard formatters.

The XML support is most complex. In Python 1.5.2, the built-in support was limited to the **xmllib** parser, which is pretty similar to the **sgmllib** module (with one important difference; **xmllib** actually tries to support the entire XML standard).

Python 2.0 comes with more advanced XML tools, based on the optional **expat** parser.

Configuration Files

The **ConfigParser** module reads and writes a simple configuration file format, similar to Windows INI files.

The **netrc** module reads **.netrc** configuration files, and the **shlex** module can be used to read any configuration file using a shell script-like syntax.

Archive Formats

Python’s standard library also provides support for the popular GZIP and ZIP (2.0 only) formats. The **gzip** module can read and write GZIP files, and the **zipfile** reads and writes ZIP files. Both modules depend on the **zlib** data compression module.

Contents

The xmllib module

The xml.parsers.expat module

The sgmllib module

The htmllib module

The htmlentitydefs module

The formatter module

The ConfigParser module

The netrc module

The shlex module

The zipfile module

The gzip module