20.1. `HTMLParser` — Simple HTML and XHTML parser¶

Note

The `HTMLParser` module has been renamed to `html.parser` in Python 3.0. The *2to3* tool will automatically adapt imports when converting your sources to 3.0.

New in version 2.2.

This module defines a class `HTMLParser` which serves as the basis for parsing text files formatted in HTML (HyperText Mark-up Language) and XHTML. Unlike the parser in `htmllib`, this parser is not based on the SGML parser in `sgmllib`.

class `HTMLParser.HTMLParser`¶

The `HTMLParser` class is instantiated without arguments.

An `HTMLParser` instance is fed HTML data and calls handler functions when tags begin and end. The `HTMLParser` class is meant to be overridden by the user to provide a desired behavior.

Unlike the parser in `htmllib`, this parser does not check that end tags match start tags or call the end-tag handler for elements which are closed implicitly by closing an outer element.

An exception is defined as well:

exception `HTMLParser.HTMLParseError`¶

Exception raised by the `HTMLParser` class when it encounters an error while parsing. This exception provides three attributes: `msg` is a brief message explaining the error, `lineno` is the number of the line on which the broken construct was detected, and `offset` is the number of characters into the line at which the construct starts.

`HTMLParser` instances have the following methods:

`HTMLParser.reset()`¶

Reset the instance. Loses all unprocessed data. This is called implicitly at instantiation time.

`HTMLParser.feed(`*data*`)`¶

Feed some text to the parser. It is processed insofar as it consists of complete elements; incomplete data is buffered until more data is fed or `close()` is called.

`HTMLParser.close()`¶

Force processing of all buffered data as if it were followed by an end-of-file mark. This method may be redefined by a derived class to define additional processing at the end of the input, but the redefined version should always call the `HTMLParser` base class method `close()`.

`HTMLParser.getpos()`¶

Return current line number and offset.

`HTMLParser.get_starttag_text()`¶

Return the text of the most recently opened start tag. This should not normally be needed for structured processing, but may be useful in dealing with HTML “as deployed” or for re-generating input with minimal changes (whitespace between attributes can be preserved, etc.).

`HTMLParser.handle_starttag(`*tag*, *attrs*`)`¶

This method is called to handle the start of a tag. It is intended to be overridden by a derived class; the base class implementation does nothing.

The *tag* argument is the name of the tag converted to lower case. The *attrs* argument is a list of `(name, value)` pairs containing the attributes found inside the tag’s <> brackets. The *name* will be translated to lower case, and quotes in the *value* have been removed, and character and entity references have been replaced. For instance, for the tag `<A HREF="http://www.cwi.nl/">`, this method would be called as `handle_starttag('a', [('href', 'http://www.cwi.nl/')])`.

Changed in version 2.6: All entity references from `htmlentitydefs` are now replaced in the attribute values.

`HTMLParser.handle_startendtag(`*tag*, *attrs*`)`¶

Similar to `handle_starttag()`, but called when the parser encounters an XHTML-style empty tag (`<a .../>`). This method may be overridden by subclasses which require this particular lexical information; the default implementation simple calls `handle_starttag()` and `handle_endtag()`.

`HTMLParser.handle_endtag(`*tag*`)`¶

This method is called to handle the end tag of an element. It is intended to be overridden by a derived class; the base class implementation does nothing. The *tag* argument is the name of the tag converted to lower case.

`HTMLParser.handle_data`(*data*)¶

This method is called to process arbitrary data. It is intended to be overridden by a derived class; the base class implementation does nothing.

`HTMLParser.handle_charref`(*name*)¶

This method is called to process a character reference of the form `&#ref;`. It is intended to be overridden by a derived class; the base class implementation does nothing.

`HTMLParser.handle_entityref`(*name*)¶

This method is called to process a general entity reference of the form `&name;` where *name* is an general entity reference. It is intended to be overridden by a derived class; the base class implementation does nothing.

`HTMLParser.handle_comment`(*data*)¶

This method is called when a comment is encountered. The *comment* argument is a string containing the text between the `--` and `--` delimiters, but not the delimiters themselves. For example, the comment `<!--text-->` will cause this method to be called with the argument `'text'`. It is intended to be overridden by a derived class; the base class implementation does nothing.

`HTMLParser.handle_decl`(*decl*)¶

Method called when an SGML declaration is read by the parser. The *decl* parameter will be the entire contents of the declaration inside the `<!`...`>` markup. It is intended to be overridden by a derived class; the base class implementation does nothing.

`HTMLParser.handle_pi`(*data*)¶

Method called when a processing instruction is encountered. The *data* parameter will contain the entire processing instruction. For example, for the processing instruction `<?proc color='red'>`, this method would be called as `handle_pi("proc color='red'")`. It is intended to be overridden by a derived class; the base class implementation does nothing.

Note

The `HTMLParser` class uses the SGML syntactic rules for processing instructions. An XHTML processing instruction using the trailing `'?'` will cause the `'?'` to be included in *data*.

20.1.1. Example HTML Parser Application¶

As a basic example, below is a very basic HTML parser that uses the `HTMLParser` class to print out tags as they are encountered:

```
from HTMLParser import HTMLParser

class MyHTMLParser(HTMLParser):

   def handle_starttag(self, tag, attrs):
       print "Encountered the beginning of a %s tag" % tag

   def handle_endtag(self, tag):
       print "Encountered the end of a %s tag" % tag
```

Table Of Contents

Previous topic

20. Structured Markup Processing Tools

Next topic

20.2. `sgmllib` — Simple SGML parser

This Page

- Show Source

Navigation