Модуль html.parser в Python

Си затор HTML

Мо класс HTMLParser, который служит основой для парсинга файлов, отформатированных в HT age) и XHTML.

Со

- Базовый пример синтаксического анализа HTML.
- Класс HTMLParser.
- Методы класса HTMLParser.
- Расширенные примеры использования парсера HTML-разметки.

Базовый пример синтаксического анализа HTML.

Ниже представлен пример простого HTML парсера, который использует класс html.parser.HTMLParser, чтобы печатать начальный и конечный тэги, а так же данные между ними:

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print("Encountered a start tag:", tag)

    def handle_endtag(self, tag):
        print("Encountered an end tag :", tag)

    def handle_data(self, data):
        print("Encountered some data  :", data)

parser = MyHTMLParser()
parser.feed('<html><head><title>Test</title></head>'
            '<body><h1>Parse me!</h1></body></html>')
```

Вывод будет следующим:

```
Encountered a start tag: html
Encountered a start tag: head
Encountered a start tag: title
Encountered some data  : Test
Encountered an end tag : title
Encountered an end tag : head
Encountered a start tag: body
Encountered a start tag: h1
Encountered some data  : Parse me!
Encountered an end tag : h1
Encountered an end tag : body
Encountered an end tag : html
```

Кл HTMLParser.

Вверх

Синтаксис:

```
from html.parser import HTMLParser

parser = HTMLParser(*, convert_charrefs=True)
```


Оп

Кл мпляр парсера, который способен проанализировать HTML-разметку, а так же некоторую не

Ес s имеет значение True (по умолчанию), то все ссылки на символы, такие как > (кроме те ментах script/style) автоматически преобразуются в соответствующие символы Unicode.

Эк HTML-данные и вызывает соответствующие методы обработчика, когда встречаются на , текст, комментарии и другие элементы разметки. Пользователь должен **создать подкласс HT** го методы для реализации желаемого поведения.

Си основе класса HTMLParser не будет проверять соответствие конечных тегов начальным те аботчик конечных тегов для элементов, которые закрываются неявно путем закрытия вн img ... />.

M MLParser.

У экземпляров HTMLParser есть следующие методы:

- HTMLParser.feed() передает текст в синтаксический анализатор,
- HTMLParser.close() обрабатывает все буферизованные данные,
- HTMLParser.reset() сбрасывает экземпляр на начальное состояние,
- HTMLParser.getpos() возвращает номер текущей строки и смещение,
- HTMLParser.get_starttag_text() текст последнего открытого начального тега,
- Методы, для переопределения в подклассе:
  - HTMLParser.handle_starttag() вызывается для открывающего HTML-тега,
  - HTMLParser.handle_endtag() вызывается для закрывающего HTML-тега,
  - HTMLParser.handle_startendtag() вызывается для HTML-тегов, которые не закрываются,
  - HTMLParser.handle_data() вызывается для текста, расположенного внутри HTML-тега,
  - HTMLParser.handle_entityref() вызывается для обработки HTML-мнемоник,
  - HTMLParser.handle_charref() вызывается для обработки ссылок на символы Юникода,
  - HTMLParser.handle_comment() вызывается при обнаружении комментария,
  - HTMLParser.handle_decl() вызывается для обработки DOCTYPE,
  - HTMLParser.handle_pi() вызывается при обнаружении инструкции по обработке,
  - HTMLParser.unknown_decl() вызывается при обнаружении неизвестного объявления,

---

HTMLParser.feed(data):

Метод HTMLParser.feed() передает текст data в синтаксический анализатор. Они обрабатывается, так как состоят из полных элементов. Неполные данные буферизуются до тех пор, пока не будет подано больше данных или не будет вызван метод HTMLParser.close(). Данные data должны быть str.

HTMLParser.close():

Метод HTMLParser.close() принудительно обрабатывает все буферизованные данные, как если бы за ними следовала метка конца файла. Этот метод может быть переопределен производным классом для определения дополнительной обработки в конце ввода, но переопределенная версия всегда должна вызывать метод базового класса HTMLParser.close().

HTMLParser.reset():

Метод HTMLParser.reset() сбрасывает экземпляр на начальное состояние. Теряет все необработанные данные. Этот метод вызывается неявно во время создания экземпляра.

HTMLParser.getpos():

Метод HTMLParser.getpos() возвращает номер текущей строки и смещение.

HT Вверх ser.get_starttag_text():

Метод HTMLParser.get_starttag_text() возвращает текст последнего открытого начального тега. Обычно это не требуется для структурированной обработки, но может быть полезно при работе с HTML *в развернутом виде* или для повторной генерации ввода с минимальными изменениями (можно сохранить пробелы между атрибутами и т. д.).

Внызываются, когда встречаются элементы данных или разметки, они предназначены для пе Реализации базового класса ничего не делают (кроме HTMLParser.handle_starttag()):

HT rttag(tag, attrs):

Me tag() метод вызывается для обработки начала тега (например,).

Ap преобразованное в нижний регистр. Аргумент attrs представляет собой список пар (name, va найденные внутри скобок тега <>. Имя тега name будет переведено в нижний регистр, ка удалены, а ссылки на HTML-сущности заменены.

На ttps://www.cwi.nl/">, этот метод будет вызываться с аргументами handle_starttag('a', [(/')]).

Вс з html.entities заменяются в значениях атрибутов.

HT tag(tag):

Метод HTMLParser.handle_endtag() вызывается для обработки конечного тега элемента (например,).

Аргумент tag - это имя тега, преобразованное в нижний регистр.

HTMLParser.handle_startendtag(tag, attrs):

Метод HTMLParser.handle_startendtag() ведет себя подобно методу HTMLParser.handle_starttag(), но вызывается, когда анализатор обнаруживает пустой тег в стиле XHTML ().

Этот метод может быть переопределен подклассами, которым требуется эта конкретная лексическая информация. Реализация по умолчанию просто вызывает HTMLParser.handle_starttag() и HTMLParser.handle_endtag().

HTMLParser.handle_data(data):

Метод HTMLParser.handle_data() вызывается для обработки произвольных данных. Например, текстовых узлов и содержимого <script>...</script> и <style>...</style>.

HTMLParser.handle_entityref(name):

Метод HTMLParser.handle_entityref() вызывается для обработки именованной ссылки на символ, вида &name; (например, >), где name является общей ссылкой на сущность (например, "gt").

Этот метод никогда не вызывается, если значение аргумента convert_charrefs конструктора HTMLParser установлено как True.

HTMLParser.handle_charref(name):

Метод HTMLParser.handle_charref() вызывается для обработки десятичных и шестнадцатеричных ссылок на числовые символы вида &#NNN; и &#xNNN;. Например, десятичный эквивалент для > равен >, тогда как шестнадцатеричный - >. В этом случае метод получит " 62 " или "x3E".

Этот метод никогда не вызывается, если значение аргумента convert_charrefs конструктора HTMLParser установлено как True.

HTMLParser.handle_comment(data):

Метод HTMLParser.handle_comment() вызывается при обнаружении комментария (например,<! --комментарий-->).

Например, комментарий <!-- comment --> вызовет этот метод с аргументом data, равным "comment".

Содержимое условных комментариев Internet Explorer (condcoms) также будет отправлено этому методу, например для <!-[if IE 9]>IE9-specific content<![endif]-->, метод получит '[if IE 9]>IE9-specific content<![endif]'.

HTMLParser.handle_decl(decl):

Me Вверх Parser.handle_decl() вызывается для обработки объявления типа документа HTML (например, <!DOCTYPE html>).

Аргументом decl будет все содержимое объявления внутри разметки! Например, 'DOCTYPE html'.

HTMLParser.handle_pi(data):

Метод HTMLParser.handle_pi() вызываемый при обнаружении инструкции по обработке. Аргумент data будет содержать всю ин

На обработке <?proc color='красный'> этот метод будет вызываться с аргументами ра 'красный'"). Метод предназначен для переопределения производным классом, реализация ба ает.

Пр использует синтаксические правила SGML для обработки инструкций. Команда обработки XH ого '?' приведет к включению '?' в аргумент data.

HT cl(data):

Ме () вызывается, когда синтаксический анализатор считывает нераспознанное объявление.

Ар все содержимое объявления внутри разметки <![...]>. Иногда бывает полезно пе очернем классе. Реализация базового класса ничего не делает.

Р ры использования парсера HTML-разметки.

Класс MyHTMLParser реализует парсер, который будет использоваться для иллюстрации дополнительных расширенных примеров:

```
from html.parser import HTMLParser
from html.entities import name2codepoint

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print("Start tag:", tag)
        for attr in attrs:
            print("     attr:", attr)

    def handle_endtag(self, tag):
        print("End tag  :", tag)

    def handle_data(self, data):
        print("Data     :", data)

    def handle_comment(self, data):
        print("Comment  :", data)

    def handle_entityref(self, name):
        c = chr(name2codepoint[name])
        print("Named ent:", c)

    def handle_charref(self, name):
        if name.startswith('x'):
            c = chr(int(name[1:], 16))
        else:
            c = chr(int(name))
        print("Num ent  :", c)

    def handle_decl(self, data):
        print("Decl     :", data)

# создаем объект парсера
>>> parser = MyHTMLParser()
```

Разбор DOCTYPE:

```
>>> parser.feed('<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">'
# Decl   : DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
```

Вверх

Разбор элемента с несколькими атрибутами и заголовком:

```
>>> parser.feed('<img src="python-logo.png" alt="The Python logo">')
# Start tag: img
#      attr: ('src', 'python-logo.png')
#      attr: ('alt', 'The Python logo')
```



```
>>> parser.feed('/h1>')
#
#
#
```

Со и элементов style, возвращается как есть, без дальнейшего анализа:

```
#
>> ="text/css">#python { color: green }</style>')
#
# css')
# : green }
#


#
>> e="text/javascript">alert("<strong>hello!</strong>");</script>')
#
#      attr: ('type', 'text/javascript')
# Data     : alert("<strong>hello!</strong>");
# End tag  : script
```

Разбор HTML-комментариев:

```
>>> parser.feed('<!-- a comment --> <!--[if IE 9]>IE-specific content<![endif]-->')
# Comment  :  a comment
# Comment  : [if IE 9]>IE-specific content<![endif]
```

Разбор HTML-мнемоник и числовых ссылок на символы Юникода и преобразование их в человеческие символы (примечание: все эти 3 ссылки эквивалентны символу '>'):

```
>>> parser.feed('&gt;&#62;&#x3E;')
# Named ent: >
# Num ent  : >
# Num ent  : >
```

Подача неполных фрагментов в метод parser.feed() работает, но [parser.handle_data()](#HTMLParser.handle_data) может вызываться более одного раза (если только значениеconvert_charrefs=True`):

```
>>> for chunk in ['<sp', 'an>buff', 'ered ', 'text</s', 'pan>']:
...     parser.feed(chunk)

# Start tag: span
# Data     : buff
# Data     : ered
# Data     : text
# End tag  : span
```

Парсинг не валидной HTML-разметки также работает (например атрибуты без кавычек):

```
>>> parser.feed('<p><a class=link href=#main>tag soup</p ></a>')
# Start tag: p
# Start tag: a
#      attr: ('class', 'link')
#      attr: ('href', '#main')
# Data     : tag soup
# End tag  : p
# End tag  : a
```