zope.structuredtext Documentation

Release 3.5.0

Zope Foundation and Contributors

February 03, 2015

Contents

1 Using Structured Text **1**

2 Customizing the document processor **3**
2.1 `zope.structuredtext` API 3

3 Indices and tables **9**

Python Module Index **11**

CHAPTER 1

Using Structured Text

The goal of StructuredText is to make it possible to express structured text using a relatively simple plain text format. Simple structures, like bullets or headings are indicated through conventions that are natural, for some definition of "natural". Hierarchical structures are indicated through indentation. The use of indentation to express hierarchical structure is inspired by the Python programming language.

Use of StructuredText consists of one to three logical steps. In the first step, a text string is converted to a network of objects using the `structurize()` facility, as in the following example:

```
raw = open("mydocument.txt").read()
from zope.structuredtext.stng import structurize
st = structurize(raw)
```

The output of `structurize()` is simply a `StructuredTextDocument` object containing `StructuredTextParagraph` objects arranged in a hierarchy. Paragraphs are delimited by strings of two or more whitespace characters beginning and ending with newline characters. Hierarchy is indicated by indentation. The indentation of a paragraph is the minimum number of leading spaces in a line containing non-white-space characters after converting tab characters to spaces (assuming a tab stop every eight characters).

`StructuredTextNode` objects support the read-only subset of the Document Object Model (DOM) API. It should be possible to process `StructuredTextNode` hierarchies using XML tools such as XSLT.

The second step in using StructuredText is to apply additional structuring rules based on text content. A variety of differentText rules can be used. Typically, these are used to implement a structured text language for producing documents, but any sort of structured text language could be implemented in the second step. For example, it is possible to use StructuredText to implement structured text formats for representing structured data. The second step, which could consist of multiple processing steps, is performed by processing, or "coloring", the hierarchy of generic StructuredTextParagraph objects into a network of more specialized objects. Typically, the objects produced should also implement the DOM API to allow processing with XML tools.

A document processor is provided to convert a StructuredTextDocument object containing only StructuredTextParagraph objects into a StructuredTextDocument object containing a richer collection of objects such as bullets, headings, emphasis, and so on using hints in the text. Hints are selected based on conventions of the sort typically seen in electronic mail or news-group postings. It should be noted, however, that these conventions are somewhat culturally dependent, fortunately, the document processor is easily customized to implement alternative rules. Here's an example of using the DOC processor to convert the output of the previous example:

```
from zope.structuredtext.document import Document
doc = Document()(st)
```

The final step is to process the colored networks produced from the second step to produce additional outputs. The final step could be performed by Python programs, or by XML tools. A Python outputter is provided for the document processor output that produces Hypertext Markup Language (HTML) text:

```
from zope.structuredtext.html import HTML
html = HTML()(doc)
```

CHAPTER 2

Customizing the document processor

The document processor is driven by two tables. The first table, named `paragraph_types`, is a sequence of callable objects or method names for coloring paragraphs. If a table entry is a string, then it is the name of a method of the document processor to be used. For each input paragraph, the objects in the table are called until one returns a value (not 'None'). The value returned replaces the original input paragraph in the output. If none of the objects in the paragraph types table return a value, then a copy of the original paragraph is used. The new object returned by calling a paragraph type should implement the `ReadOnlyDOM`, `StructuredTextColorizable`, and `StructuredTextSubparagraphContainer` interfaces. See the `zope.structuredtext.document` source file for examples.

A paragraph type may return a list or tuple of replacement paragraphs, this allowing a paragraph to be split into multiple paragraphs.

The second table, `text_types`, is a sequence of callable objects or method names for coloring text. The callable objects in this table are used in sequence to transform the input text into new text or objects. The callable objects are passed a string and return nothing (`None`) or a three-element tuple consisting of:

- a replacement object,
- a starting position, and
- an ending position

The text from the starting position is (logically) replaced with the replacement object. The replacement object is typically an object that implements that implements the `ReadOnlyDOM` and `StructuredTextColorizable` interfaces. The replacement object can also be a string or a list of strings or objects. Replacement is done from beginning to end and text after the replacement ending position will be passed to the character type objects for processing.

Contents:

2.1 `zope.structuredtext` API

2.1.1 `zope.structuredtext.document`

Structured text document parser

class `zope.structuredtext.document`**.Document**

Class instance calls [ex.=> x()] require a structured text structure. Doc will then parse each paragraph in the structure and will find the special structures within each paragraph. Each special structure will be stored as an instance. Special structures within another special structure are stored within the 'top' structure EX : '-underline this-' => would be turned into an underline instance. '-underline **this**' would be stored as an underline instance with a strong instance stored in its string

color_text (*text*, *types=None*)
Search the paragraph for each special structure

doc_sgml (*s*, *expr=<built-in method search of _sre.SRE_Pattern object at 0x7f1b5d56ee00>*)
SGML text is ignored and outputed as-is

parse (*raw_string*, *text_type*, *type=<type 'type'>*)
Parse accepts a raw_string, an expr to test the raw_string, and the raw_string's subparagraphs.

Parse will continue to search through raw_string until all instances of expr in raw_string are found.

If no instances of expr are found, raw_string is returned. Otherwise a list of substrings and instances is returned

class zope.structuredtext.document.**DocumentWithImages**
Document with images

2.1.2 zope.structuredtext.stletters

Structured text character classes

2.1.3 zope.structuredtext.stng

Core document model.

class zope.structuredtext.stng.**StructuredTextBullet** (*src*, *subs=None*, ***kw*)
Represents a section of a document with a title and a body

class zope.structuredtext.stng.**StructuredTextColumn** (*text*, *span*, *align*, *valign*, *typ*, *kw*)
StructuredTextColumn is a cell/column in a table. A cell can hold multiple paragraphs. The cell is either classified as a StructuredTextTableHeader or StructuredTextTableData.

class zope.structuredtext.stng.**StructuredTextDescription** (*title*, *src*, *subs*, ***kw*)
Represents a section of a document with a title and a body

class zope.structuredtext.stng.**StructuredTextDescriptionBody** (*src*, *subs=None*, ***kw*)
Represents a section of a document with a title and a body

class zope.structuredtext.stng.**StructuredTextDescriptionTitle** (*src*, *subs=None*, ***kw*)
Represents a section of a document with a title and a body

class zope.structuredtext.stng.**StructuredTextDocument** (*subs=None*, ***kw*)
A StructuredTextDocument holds StructuredTextParagraphs as its subparagraphs.

class zope.structuredtext.stng.**StructuredTextExample** (*subs*, ***kw*)
Represents a section of document with literal text, as for examples

class zope.structuredtext.stng.**StructuredTextImage** (*value*, ***kw*)
A simple embedded image

class zope.structuredtext.stng.**StructuredTextNumbered** (*src*, *subs=None*, ***kw*)
Represents a section of a document with a title and a body

class zope.structuredtext.stng.**StructuredTextSection** (*src*, *subs=None*, ***kw*)
Represents a section of a document with a title and a body

class zope.structuredtext.stng.**StructuredTextSectionTitle** (*src*, *subs=None*, ***kw*)
Represents a section of a document with a title and a body

class zope.structuredtext.stng.**StructuredTextTable**(*rows*, *src*, *subs*, ***kw*)
rows is a list of lists containing tuples, which represent the columns/cells in each rows. EX rows = [[('row 1:column1',1)],[('row2:column1',1)]]

getColorizableTexts()
return a tuple where each item is a column/cell's contents. The tuple, result, will be of this format. ("r1 col1", "r1=col2", "r2 col1", "r2 col2")

setColorizableTexts(*texts*)
texts is going to a tuple where each item is the result of being mapped to the colortext function. Need to insert the results appropriately into the individual columns/cells

zope.structuredtext.stng.**display**(*struct*)
Runs through the structure and prints out the paragraphs. If the insertion works correctly, display's results should mimic the orignal paragraphs.

zope.structuredtext.stng.**display2**(*struct*)
Runs through the structure and prints out the paragraphs. If the insertion works correctly, display's results should mimic the orignal paragraphs.

zope.structuredtext.stng.**findlevel**(*levels*, *indent*)
Remove all level information of levels with a greater level of indentation. Then return which level should insert this paragraph

zope.structuredtext.stng.**indention**(*str*, *front=<built-in method match of _sre.SRE_Pattern object at 0x7f1b5d13b210>*)
Find the number of leading spaces. If none, return 0.

zope.structuredtext.stng.**insert**(*struct*, *top*, *level*)
Find what will be the parant paragraph of a sentence and return that paragraph's sub-paragraphs. The new paragraph will be appended to those sub-paragraphs

zope.structuredtext.stng.**structurize**(*paragraphs*, *delimiter=<_sre.SRE_Pattern object at 0x7f1b5d570648>*)
Accepts paragraphs, which is a list of lines to be parsed. structurize creates a structure which mimics the structure of the paragraphs. Structure => [paragraph,[sub-paragraphs]]

2.1.4 zope.structuredtext.stdom

DOM implementation in StructuredText: read-only methods

class zope.structuredtext.stdom.**Attr**(*name*, *value*, *specified=1*)
Attr interface - The Attr interface represents an attriubte in an Element object. Attr objects inherit the Node Interface

getName()
Returns the name of this attribute.

getNodeName()
The name of this node, depending on its type

getNodeType()
A code representing the type of the node.

getNodeValue()
The value of this node, depending on its type

getSpecified()
If this attribute was explicitly given a value in the original document, this is true; otherwise, it is false.

class zope.structuredtext.stdom.**Element**
Element interface

getAttribute(*name*)
Retrieves an attribute value by name.

getAttributeNode(*name*)
Retrieves an Attr node by name or None if there is no such attribute.

getElementsByTagName(*tagname*)
Returns a NodeList of all the Elements with a given tag name in the order in which they would be encountered in a preorder traversal of the Document tree. Parameter: tagname The name of the tag to match (* = all tags). Return Value: A new NodeList object containing all the matched Elements.

getNodeName()
The name of this node, depending on its type

getNodeType()
A code representing the type of the node.

getParentNode()
The parent of this node. All nodes except Document DocumentFragment and Attr may have a parent

getTagName()
The name of the element

class zope.structuredtext.stdom.**NamedNodeMap**(*data=None*)
NamedNodeMap interface - Is used to represent collections of nodes that can be accessed by name. NamedNodeMaps are not maintained in any particular order.

Python extensions: can use sequence-style 'len', 'getitem', and 'for..in' constructs, and mapping-style 'getitem'.

getLength()
The length of the NodeList

getNamedItem(*name*)
Retrieves a node specified by name. Parameters: name Name of a node to retrieve. Return Value A Node (of any type) with the specified name, or None if the specified name did not identify any node in the map.

item(*index*)
Returns the index-th item in the map

class zope.structuredtext.stdom.**Node**
Node Interface

getAttributes()
Returns a NamedNodeMap containing the attributes of this node (if it is an element) or None otherwise.

getChildren()
Get a Python sequence of children

getNextSibling()
The node immediately preceding this node. If there is no such node, this returns None.

getNodeName()
The name of this node, depending on its type

getNodeValue()
The value of this node, depending on its type

getOwnerDocument()
The Document object associated with this node, if any.

getParentNode()
The parent of this node. All nodes except Document DocumentFragment and Attr may have a parent

getPreviousSibling()
The node immediately preceding this node. If there is no such node, this returns None.

hasChildNodes()
Returns true if the node has any children, false if it doesn't.

class zope.structuredtext.stdom.**NodeList**(*list=None*)
NodeList interface - Provides the abstraction of an ordered collection of nodes.

Python extensions: can use sequence-style 'len', 'getitem', and 'for..in' constructs.

getLength()
The length of the NodeList

item(*index*)
Returns the index-th item in the collection

class zope.structuredtext.stdom.**NodeWrapper**(*aq_self*, *aq_parent*)
This is an acquisition-like wrapper that provides parent access for DOM sans circular references!

getNextSibling()
The node immediately preceding this node. If there is no such node, this returns None.

getOwnerDocument()
The Document object associated with this node, if any.

getParentNode()
The parent of this node. All nodes except Document DocumentFragment and Attr may have a parent

getPreviousSibling()
The node immediately preceding this node. If there is no such node, this returns None.

class zope.structuredtext.stdom.**ParentNode**
A node that can have children, or, more precisely, that implements the child access methods of the DOM.

getChildNodes(*type=<type 'type'>*, *sts=(<type 'unicode'>, <type 'str'>)*)
Returns a NodeList that contains all children of this node. If there are no children, this is a empty NodeList

getFirstChild(*type=<type 'type'>*, *sts=(<type 'unicode'>, <type 'str'>)*)
The first child of this node. If there is no such node this returns None

getLastChild(*type=<type 'type'>*, *sts=(<type 'unicode'>, <type 'str'>)*)
The last child of this node. If there is no such node this returns None.

2.1.5 zope.structuredtext.html

HTML renderer for STX documents.

2.1.6 zope.structuredtext.docbook

Render STX document as docbook.

class zope.structuredtext.docbook.**DocBook**
Structured text document renderer for Docbook.

2.1.7 zope.structuredtext

Zope structured text markeup

Consider the following example:

```
>>> from zope.structuredtext.stng import structurize
>>> from zope.structuredtext.document import DocumentWithImages
>>> from zope.structuredtext.html import HTMLWithImages
>>> from zope.structuredtext.docbook import DocBook

We first need to structurize the string and make a full-blown
document out of it:

>>> structured_string = '''
... Title Here
...
...     Body text here.'''
>>> struct = structurize(structured_string)
>>> doc = DocumentWithImages()(struct)

Now feed it to some output generator, in this case HTML or DocBook:

>>> output = HTMLWithImages()(doc, level=1)
>>> output = DocBook()(doc, level=1)
```

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

Z

zope.structuredtext, 8
zope.structuredtext.docbook, 7
zope.structuredtext.document, 3
zope.structuredtext.html, 7
zope.structuredtext.stdom, 5
zope.structuredtext.stletters, 4
zope.structuredtext.stng, 4

A

Attr (class in zope.structuredtext.stdom), 5

C

color_text() (zope.structuredtext.document.Document method), 3

D

display() (in module zope.structuredtext.stng), 5
display2() (in module zope.structuredtext.stng), 5
doc_sgml() (zope.structuredtext.document.Document method), 4
DocBook (class in zope.structuredtext.docbook), 7
Document (class in zope.structuredtext.document), 3
DocumentWithImages (class in zope.structuredtext.document), 4

E

Element (class in zope.structuredtext.stdom), 5

F

findlevel() (in module zope.structuredtext.stng), 5

G

getAttribute() (zope.structuredtext.stdom.Element method), 6
getAttributeNode() (zope.structuredtext.stdom.Element method), 6
getAttributes() (zope.structuredtext.stdom.Node method), 6
getChildNodes() (zope.structuredtext.stdom.ParentNode method), 7
getChildren() (zope.structuredtext.stdom.Node method), 6
getColorizableTexts() (zope.structuredtext.stng.StructuredTextTable method), 5
getElementsByTagName() (zope.structuredtext.stdom.Element method), 6
getFirstChild() (zope.structuredtext.stdom.ParentNode method), 7
getLastChild() (zope.structuredtext.stdom.ParentNode method), 7
getLength() (zope.structuredtext.stdom.NamedNodeMap method), 6
getLength() (zope.structuredtext.stdom.NodeList method), 7
getName() (zope.structuredtext.stdom.Attr method), 5
getNamedItem() (zope.structuredtext.stdom.NamedNodeMap method), 6
getNextSibling() (zope.structuredtext.stdom.Node method), 6
getNextSibling() (zope.structuredtext.stdom.NodeWrapper method), 7
getNodeName() (zope.structuredtext.stdom.Attr method), 5
getNodeName() (zope.structuredtext.stdom.Element method), 6
getNodeName() (zope.structuredtext.stdom.Node method), 6
getNodeType() (zope.structuredtext.stdom.Attr method), 5
getNodeType() (zope.structuredtext.stdom.Element method), 6
getNodeValue() (zope.structuredtext.stdom.Attr method), 5
getNodeValue() (zope.structuredtext.stdom.Node method), 6
getOwnerDocument() (zope.structuredtext.stdom.Node method), 6
getOwnerDocument() (zope.structuredtext.stdom.NodeWrapper method), 7
getParentNode() (zope.structuredtext.stdom.Element method), 6
getParentNode() (zope.structuredtext.stdom.Node method), 6
getParentNode() (zope.structuredtext.stdom.NodeWrapper method), 7
getPreviousSibling() (zope.structuredtext.stdom.Node method), 7
getPreviousSibling() (zope.structuredtext.stdom.NodeWrapper method), 7

getSpecified() (zope.structuredtext.stdom.Attr method), 5
getTagName() (zope.structuredtext.stdom.Element method), 6

H

hasChildNodes() (zope.structuredtext.stdom.Node method), 7

I

indention() (in module zope.structuredtext.stng), 5
insert() (in module zope.structuredtext.stng), 5
item() (zope.structuredtext.stdom.NamedNodeMap method), 6
item() (zope.structuredtext.stdom.NodeList method), 7

N

NamedNodeMap (class in zope.structuredtext.stdom), 6
Node (class in zope.structuredtext.stdom), 6
NodeList (class in zope.structuredtext.stdom), 7
NodeWrapper (class in zope.structuredtext.stdom), 7

P

ParentNode (class in zope.structuredtext.stdom), 7
parse() (zope.structuredtext.document.Document method), 4

S

setColorizableTexts() (zope.structuredtext.stng.StructuredTextTable method), 5
StructuredTextBullet (class in zope.structuredtext.stng), 4
StructuredTextColumn (class in zope.structuredtext.stng), 4
StructuredTextDescription (class in zope.structuredtext.stng), 4
StructuredTextDescriptionBody (class in zope.structuredtext.stng), 4
StructuredTextDescriptionTitle (class in zope.structuredtext.stng), 4
StructuredTextDocument (class in zope.structuredtext.stng), 4
StructuredTextExample (class in zope.structuredtext.stng), 4
StructuredTextImage (class in zope.structuredtext.stng), 4
StructuredTextNumbered (class in zope.structuredtext.stng), 4
StructuredTextSection (class in zope.structuredtext.stng), 4
StructuredTextSectionTitle (class in zope.structuredtext.stng), 4
StructuredTextTable (class in zope.structuredtext.stng), 4
structurize() (in module zope.structuredtext.stng), 5

Z

zope.structuredtext (module), 8
zope.structuredtext.docbook (module), 7
zope.structuredtext.document (module), 3
zope.structuredtext.html (module), 7
zope.structuredtext.stdom (module), 5
zope.structuredtext.stletters (module), 4
zope.structuredtext.stng (module), 4