sphinxcontrib.datatemplates Documentation

Release 0.0.0

Doug Hellmann

Jan 16, 2020

Contents

`sphinxcontrib.datatemplates` helps you use static data in machine readable format in your documentation by letting you define Jinja2 templates to turn JSON, YAML, XML, or CSV data into reStructuredText for Sphinx to render as part of its output.

- Repo: https://github.com/sphinxcontrib/sphinxcontrib.datatemplates
- Docs: http://sphinxcontribdatatemplates.readthedocs.io/

CHAPTER 1

Using datatemplate

The `datatemplate` directive is the interface between the data source and the rendering template. It requires two parameters.

`.. datatemplate:json::`
: Load `source` via `json.load()`

`.. datatemplate:yaml::`
: Load `source` via PyYAML (`yaml.safe_load`)

`.. datatemplate:xml::`
: Load `source` via `xml.etree.ElementTree.parse()` (actually using `defusedxml`)

`.. datatemplate:csv::`
: Load `source` via `csv.reader()` or `csv.DictReader` depending on `header`

1.1 Template Context

When a `datatemplate` directive is processed, the data is passed to the template through its context so that the symbol `data` is available as a global variable.

Important: The data is loaded from the source and passed directly to the template. No pre-processing is done on the data, so the template needs to handle aspects like `None` values and fields that have values that may interfere with parsing reStructuredText.

1.2 Template Helpers

These helper functions are exposed using their short name (without the module prefix) in the template context.

sphinxcontrib.datatemplates.helpers.**make_list_table**(*headers*, *data*, *title=''*, *columns=None*)

Build a list-table directive.

Parameters

- **headers** – List of header values.
- **data** – Iterable of row data, yielding lists or tuples with rows.
- **title** – Optional text to show as the table title.
- **columns** – Optional widths for the columns.

sphinxcontrib.datatemplates.helpers.**make_list_table_from_mappings**(*headers*, *data*, *title*, *columns=None*)

Build a list-table directive.

Parameters

- **headers** – List of tuples containing header title and key value.
- **data** – Iterable of row data, yielding mappings with rows.
- **title** – Optional text to show as the table title.
- **columns** – Optional widths for the columns.

CHAPTER 2

JSON Samples

2.1 Data File

```
{
    "key1": "value1",
    "key2": [
        "list item 1",
        "list item 2",
        "list item 3"
    ],
    "nested-list": [
        ["a", "b", "c"],
        ["A", "B", "C"]
    ],
    "mapping-series": [
        {"cola": "a", "colb": "b", "colc": "c"},
        {"cola": "A", "colb": "B", "colc": "C"}
    ]
}
```

2.2 Template File

```
.. -*- mode: rst -*-

Static Heading
--------------

Individual Item
~~~~~~~~~~~~~~~

{{ data['key1'] }}
```

(continues on next page)

(continued from previous page)

```
List of Items
~~~~~~~~~~~~~

{% for item in data['key2'] %}
- {{item}}
{% endfor %}

Nested List Table
~~~~~~~~~~~~~~~~~

Rendering a table from a list of nested sequences using hard-coded
headers.

{{ make_list_table(
    ['One', 'Two', 'Three'],
    data['nested-list'],
    title='Table from nested lists',
    ) }}

Mapping Series Table
~~~~~~~~~~~~~~~~~~~~

Rendering a table from a list of nested dictionaries using dynamic
headers.

{{ make_list_table_from_mappings(
    [('One', 'cola'), ('Two', 'colb'), ('Three', 'colc')],
    data['mapping-series'],
    title='Table from series of mappings',
    ) }}
```

2.3 Rendered Output

CHAPTER 3

YAML Samples

3.1 Data File

```
---
key1: value1
key2:
  - list item 1
  - list item 2
  - list item 3
nested-list:
  - ['a', 'b', 'c']
  - ['A', 'B', 'C']
mapping-series:
  - cola: a
    colb: b
    colc: c
  - cola: A
    colb: B
    colc: C
```

3.2 Template File

```
.. -*- mode: rst -*-

Static Heading
--------------

Individual Item
~~~~~~~~~~~~~~~

{{ data['key1'] }}
```

(continues on next page)

(continued from previous page)

```
List of Items
~~~~~~~~~~~~~

{% for item in data['key2'] %}
- {{item}}
{% endfor %}

Nested List Table
~~~~~~~~~~~~~~~~~

Rendering a table from a list of nested sequences using hard-coded
headers.

{{ make_list_table(
    ['One', 'Two', 'Three'],
    data['nested-list'],
    title='Table from nested lists',
    ) }}

Mapping Series Table
~~~~~~~~~~~~~~~~~~~~

Rendering a table from a list of nested dictionaries using dynamic
headers.

{{ make_list_table_from_mappings(
    [('One', 'cola'), ('Two', 'colb'), ('Three', 'colc')],
    data['mapping-series'],
    title='Table from series of mappings',
    ) }}
```

3.3 Rendered Output

CHAPTER 4

CSV Samples

4.1 Data File

```
a         b         c
Eins         Zwei         Drei
1         2         3
I         II          III
```

4.2 Template File

```
.. -*- mode: rst -*-

Static Heading
--------------

Individual Cell in Row
~~~~~~~~~~~~~~~~~~~~~~

{{ data[0].a }}

List of Cells in Row
~~~~~~~~~~~~~~~~~~~~

{% for item in data[0].items() %}
- {{item[0]}}: {{item[1]}}
{% endfor %}

Mapping Series Table
~~~~~~~~~~~~~~~~~~~~

Rendering a table from a list of nested dictionaries using dynamic
```

(continues on next page)

(continued from previous page)

```
headers.

{{ make_list_table_from_mappings(
    [('One', 'a'), ('Two', 'b'), ('Three', 'c')],
    data,
    title='Table from series of mappings',
    ) }}
```

4.3 Code

```
.. datatemplate:csv::
    :source: sample.csv
    :template: csv-sample.tmpl
    :headers:
    :dialect: excel-tab
```

4.4 Rendered Output

CHAPTER 5

XML Samples

5.1 Data File

```
<sample>
    <key1>value1</key1>
    <key2>
        <item>list item 1</item>
        <item>list item 2</item>
        <item special='yes'>list item 3</item>
    </key2>
    <mappingseries>
        <mapping>
            <cola special='yes'>a</cola>
            <colb>b</colb>
            <colc>c</colc>
        </mapping>
        <mapping>
            <cola>A</cola>
            <colb special='yes'>B</colb>
            <colc>C</colc>
        </mapping>
    </mappingseries>
</sample>
```

5.2 Template File

```
.. -*- mode: rst -*-

Static Heading
--------------
```

(continues on next page)

(continued from previous page)

```
Individual Item
~~~~~~~~~~~~~~~

{{ data.find('key1').text }}

List of Items
~~~~~~~~~~~~~

{% for item in data.find('key2') %}
- {{item.text}}
{% endfor %}

XPath for Items
~~~~~~~~~~~~~~~

See `XPath support <https://docs.python.org/3/library/xml.etree.elementtree.html
↪#xpath-support>`_

{% for item in data.findall(".//*[@special='yes']") %}
- {{item.text}}
{% endfor %}
```

5.3 Rendered Output

CHAPTER 6

Legacy Samples

The `datatemplate` directive is should no longer be used. It is deprecated, and will be removed in the next release.

6.1 Data File

```
---
key1: value1
key2:
  - list item 1
  - list item 2
  - list item 3
nested-list:
  - ['a', 'b', 'c']
  - ['A', 'B', 'C']
mapping-series:
  - cola: a
    colb: b
    colc: c
  - cola: A
    colb: B
    colc: C
```

6.2 Template File

```
.. -*- mode: rst -*-

Static Heading
--------------

Individual Item
```

(continues on next page)

(continued from previous page)

```
~~~~~~~~~~~~~~~

{{ data['key1'] }}

List of Items
~~~~~~~~~~~~~

{% for item in data['key2'] %}
- {{item}}
{% endfor %}

Nested List Table
~~~~~~~~~~~~~~~~~

Rendering a table from a list of nested sequences using hard-coded
headers.

{{ make_list_table(
    ['One', 'Two', 'Three'],
    data['nested-list'],
    title='Table from nested lists',
    ) }}

Mapping Series Table
~~~~~~~~~~~~~~~~~~~~

Rendering a table from a list of nested dictionaries using dynamic
headers.

{{ make_list_table_from_mappings(
    [('One', 'cola'), ('Two', 'colb'), ('Three', 'colc')],
    data['mapping-series'],
    title='Table from series of mappings',
    ) }}
```

6.3 Rendered Output

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

D

M

S