copytext Documentation

Release 0.1.7 (beta)

NPR

January 27, 2015

Contents

CHAPTER 1

About

copytext is a library for accessing a spreadsheet as a native Python object suitable for templating.

On the NPR Visuals team we use this as part of our app-template. Whenever a project is rendered we fetch a Google Spreadsheet containing all the project's editable text. This spreadsheet is passed to copytext, which produces an object suitable for using in our Flask templates. This allows us to give our writers and editors a document to write in which they are more comfortable with than they would be editing the code directly.

- Repository: https://github.com/nprapps/copytext
- Issues: https://github.com/nprapps/copytext/issues
- Documentation: http://copytext.readthedocs.org/
- Visuals blog: http://blog.apps.npr.org/

CHAPTER 2

Installation

2.1 Users

If you only want to use copytext, install it this way:

```
pip install copytext
```

2.2 Developers

If you are a developer that also wants to hack on copytext, install it this way:

```
git clone git://github.com/nprapps/copytext.git
cd copytext
mkvirtualenv --no-site-packages copytext
pip install -r requirements.txt
python setup.py develop
nosetests --with-coverage --cover-package=copytext
```

CHAPTER 3

Usage

Here is an example spreadsheet:

And here is code using this data:

```
import copytext

# Instantiate our copy, this parses the XLSX workbook
copy = copytext.Copy('examples/test_copy.xlsx')

# Get a sheet named "content"
sheet = copy['content']

# The sheet has "key" and "value" columns
# This tells copytext to access the value by the key

# Print the value where the "key" is named "lorem_ipsum"
print sheet['lorem_ipsum']

# Print the value in the third row (counting headers)
print sheet[2]

# The rows themselves are also objects
row = sheet['lorem_ipsum']

# You can access the columns by indexing into the row

# Print the key column of the row
print row['key']

# Print the first column in the row
print row[0]

# You can also iterate over rows
for row in sheet:
    # Print the value
    print row

    # Print the key/value pair
    print row['key'], row['value']

# This sheet has "term" and "definition" columns, but no "key"
sheet = copy['example_list']
```

```
# This won't work
# print sheet[0]

# But this will
for row in sheet:
    print row['term'], row['definition']

# You can have as many rows and columns as you want!

# Serialize a sheet to json
js = sheet.json()

# Serialize an entire workbook to json
js = copy.json()
```

Note: Copytext only understands `xlsx` files, and all cells must be converted to text formatting. Copytext does not grok dates or numbers.

CHAPTER 4

Using with Flask

Probably the most significant use case for copytext is as an input to a template system. For example, here is how you would use it with Flask's Jinja-based templates:

Your view:

```
from flask import render_template

import copytext

@app.route('/')
def index():
    context = {
        'COPY': copytext.Copy('examples/test_copy.xlsx')
    }

    return render_template('index.html', **context)
```

And in your template:

```
<header>
    <h1>{{ COPY.content.header_title }}</h1>
    <h2>{{ COPY.content.lorem_ipsum }}</h2>
</header>

<dl>
    {% for row in COPY.example_list %}
    <dt>{{ row.term }}</dt><dd>{{ row.definition }}</dd>
    {% endfor %}
</dl>
```

copytext automatically marks all strings as safe (`Markup` in Jinja parlance).

Note: Jinja templates automatically proxy attribute access to property access, which is why you see `row.term` instead of `row['term']` in these examples. This means you can also do `row.0` to access the first column.

Need a JSON version of your copytext for the client?

```
<script type="text/javascript">
    var COPY = {{ COPY.json() }};
</script>
```

CHAPTER 5

License

The MIT License

Copyright (c) 2014 NPR

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CHAPTER 6

Changelog

6.1 0.1.7

- Maintain column order in JSON keys

6.2 0.1.6

- Explicitly convert cells to text internally. (#16)
- Support serializing a Copy or Sheet object to JSON. (#17)

6.3 0.1.5

- Properly serialize rows with keys, but not values.

6.4 0.1.4

- Skip header row when iterating over rows in a sheet.

6.5 0.1.3

- Fixes for Markup handling.

6.6 0.1.2

- Support null checking on cells. (#15)
- Integrate markupsafe and elminate cell_class_wrapper.
- Errors and rows can now test false. (#14)

6.7 0.1.1

- First draft of docs. (#4)
- Explicit handling of fields with dates/numbers. (#8)
- Tests for unicode support. (#1)

6.8 0.1.0

- Initial import.

CHAPTER 7

Indices and tables

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