collective.table Documentation

Release 1.0.1

August 29, 2011

CONTENTS

Project title collective.table

Latest version 0.3

Project page http://pypi.python.org/pypi/collective.table

Source http://github.com/nzupan/collective.table

Summary

This is a Plone 4 add-on project for storing tabular data inside Plone, replacing the "store equipment lists, book loans, etc. in Excel files and upload to Plone". It's a list of items where members can add/edit columns and rows and of course data. By default, collective.table stores it's data in the ZODB, however storage is pluggable and it's easy to write storage adapters to have data stored in SQL, LDAP, etc. It is based on DataTables and jEditable jQuery plugins.

CHAPTER

ONE

TABLE OF CONTENTS

1.1 Definitions of basic terms

DataTables A jQuery plugin to support advanced interaction controls to any HTML table. More at http://datatables.net/.

jEditable A edit-in-place jQuery plugin. More at http://www.appelsiini.net/projects/jeditable.

1.2 Developer conventions

1.2.1 Commit checklist

Before every commit you must:

- run all *Unit tests*.
- perform *Syntax validation*.
- add an entry to *Changelog* (if applicable).
- add an entry to *Sphinx documentation* (if applicable).

You can run all syntax check and all test with a single command:

```
$ ./pre-commit-check.sh
```

1.2.2 Unit tests

Un-tested code is broken code.

For every feature you add to the codebase you must also add tests for it.

You can run tests like this:

```
$ bin/test -s collective.table
```

1.2.3 Syntax validation

All Python source code should be *PEP-8* valid and checked for syntax errors. Tools for checking this are *pep8* and *pyflakes*.

If possible make your editor run *pep8* and *pyflakes* on your current file every time you save it. Useful links:

- http://github.com/ppierre/python-pep8-tmbundle
- http://www.arthurkoziel.com/2008/06/28/pyflakes-installation-and-textmate-integration/

Alternatively you can use these two commands to check style manually:

```
$ bin/pyflakes collective/table
$ bin/pep8 collective/table
```

1.2.4 Changelog

We track all feature-level changes to code inside `docs/HISTORY.txt`. Examples:

- added feature X
- removed Y
- fixed bug Z

1.2.5 Sphinx documentation

Un-documented code is broken code.

For every feature you add to the codebase you must also add documentation for it in `docs/sphinx/`.

After adding documentation, re-build *Sphinx* and check how it is displayed:

```
$ bin/sphinxbuilder
$ open docs/html/index.html
```

1.2.6 Sorting imports

As a stylistic guide: Imports of code from other modules should always be alphabetically sorted with no empty lines between imports. The only exception to this rule is to keep one empty line between a group of `from x import y` and a group of `import y` imports.

BAD

```
import os

from plone.app.testing import login
from collective.table.tests.base import TableIntegrationTestCase
```

GOOD

```
from collective.table.tests.base import TableIntegrationTestCase
from plone.app.testing import login

import os
```

1.2.7 Multiple imports

1. Don't use * to import *everything* from a module.
2. Don't use commas to import multiple stuff on a single line.

3. Don't use relative paths.

BAD

```
from collective.table.local import *
from collective.table.local import add_row, delete_rows
from .local import update_cell
```

GOOD

```
from collective.table.local import add_row
from collective.table.local import delete_rows
from collective.table.local import update_cell
```

1.3 API

1.3.1 Archetypes DataTable field and widget

1.3.2 BrowserView that displays the Table

1.3.3 Local source

1.3.4 The *Table* content-type

CHAPTER

TWO

QUICK START

If you have a Linux or OS X system already capable of running Plone then a quick start with *collective.table* is as follows:

```
$ git checkout git@github.com:nzupan/collective.table.git
$ cd collective.table/
$ virtualenv -p python2.6 --no-site-packages ./
$ bin/python bootstrap.py
$ bin/buildout
$ bin/instance fg
```

CHAPTER

THREE

CHANGELOG

3.1 1.0.1 (2011-08-22)

- Newline was missing in Future.rst. [zupo]

3.2 1.0 (2011-08-22)

- Added support for localizations. [zupo]
- More cleanup. [zupo]
- Added support for source configuration view. [MJ]
- Added project to ReadTheDocs.org. [zupo]
- Refactored row deletion handling. [MJ]
- Support for multiple tables is back. [MJ]
- Move the versions section to re-enable the mr.developer entries. [MJ]
- Refactored usage of `storage` into `source` to avoid confusing them with Archetypes storages. [zupo]
- Save source name per context, not globally on the field. [zupo]
- Split `manageable` flag into `editable`, `sortable` and `queryable` flags. [zupo]
- Added shebangs and module-level docstrings. [zupo]
- Added a script to do pre-commit checks. [zupo]
- PEP8 and PyFlakes cleanup. [zupo]
- Added Sphinx documentation. [zupo]

3.3 0.3alpha (2011-07-20)

- Fixed a bug with deleting rows. [zupo]

3.4 0.2alpha (2011-07-20)

- Renamed `README.txt` to `README.rst` and added `README` as a soft-link. [zupo]

3.5 0.1alpha (2011-07-19)

- Initial release. [zupo]

CHAPTER

FOUR

ROADMAP

4.1 Current status

At this point, collective.table offers basic functionalities and is usable. It can be used as an out-of-the-box product for end-users or as a third-party Archetypes widget for custom content-types.

4.2 Future

Being pluggable as it is, we expect more people to use it with their own content-types on their own custom data sources. Hopefully some that code will propagate back into the community so we can have multiple source possibilities in the future. UI could also use some more love.

CHAPTER

FIVE

CONTRIBUTORS

- Martijn Pieters (Jarn AS) is the original author.
- Nejc Zupan (NiteoWeb Ltd.) drove the code towards the first few releases as part of his Google Summer of Code project.

CHAPTER

SIX

INDICES AND TABLES

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