CoVE Documentation

Release beta

Open Data Services Co-operative Limited

Mar 19, 2018

Contents

Contents:

CHAPTER 1

Deployment Notes

General Django deployment considerations apply to deploying Cove. We deploy using Apache and uwsgi using this Salt State file.

1.1 Before a live deploy

Travis tests will fail if a branch isn't ready to be merged and deployed. This includes if OCDS translations are missing.

1.2 How to do a live deploy of CoVE

1.2.1 OCDS

1. Do the actual deploy. From the opendataservices-deploy directory:

```
salt-ssh --state-output=mixed -L 'cove-live-ocds-2' state.highstate
```

1. Check that the latest commit is shown in the footer of http://standard.open-contracting.org/validator/
2. Test that the live site is working as expected. From the cove directory:

```
CUSTOM_SERVER_URL=http://standard.open-contracting.org/ DJANGO_SETTINGS_MODULE=cove_
↪ocds.settings py.test cove_ocds/tests_functional.py -n 4
```

1.2.2 360Giving

1. Do the actual deploy. From the opendataservices-deploy directory:

```
salt-ssh --state-output=mixed -L 'cove-360-live' state.highstate
```

1. Check that the latest commit is shown in the footer of https://dataquality.threesixtygiving.org/

2. Test that the live site is working as expected. From the cove directory:

```
CUSTOM_SERVER_URL=https://dataquality.threesixtygiving.org/ DJANGO_SETTINGS_
↪MODULE=cove_360.settings py.test cove_360/tests_functional.py -n 4
```

1.2.3 IATI

1. Do the actual deploy. From the opendataservices-deploy directory:

```
salt-ssh --state-output=mixed -L 'cove-live-iati' state.highstate
```

1. Check that the latest commit is shown in the footer of http://iati.cove.opendataservices.coop/
2. Test that the live site is working as expected. From the cove directory:

```
CUSTOM_SERVER_URL=http://iati.cove.opendataservices.coop/ DJANGO_SETTINGS_MODULE=cove_
↪iati.settings py.test cove_iati/tests_functional.py -n 4
```

CHAPTER 2

How Cove deals with errors

Errors in Cove can be broken down into 2 categories:

- Deliberately caught errors - to some extent we're expecting something to go wrong, probably due to people's data
- Uncaught 500 errors - something unexpected breaks

Breaking these down further:

- Deliberately caught errors
  - Custom message - something's wrong with the data, and we know what, so are able to display some custom help text. Since we know this is a data problem, it doesn't get logged in Sentry.
  - Generic message - something went wrong that we think is very likely to be a data issue (e.g. conversion failed), but we don't have a custom error message for it. We show the user the caught error, but also log it to Sentry.
- Uncaught 500 errors
  - Themed 500 error page - an otherwise uncaught exception, but we successfully rendered the friendly, well themed 500 page. These are always reported to Sentry.
  - Unthemed 500 error page - something went very wrong and we couldn't even display the nice 500 error page. The error should be reported to Sentry, but that may be broken too! In general these are serious bugs, and should be reported.

In an ideal world we want to eliminate all error messages except for the custom ones (ie. the only errors are data errors, and we can tell the users how to fix them).

CHAPTER 3

Using cove as a library (cove.lib)

Currently the main user of `cove.lib` is `cove.views` which is responsible for rendering the Cove web application.

3.1 common

class cove.lib.common.**CustomJsonrefLoader**(***kwargs*)
This ref loader is only for use with the jsonref library and NOT jsonschema.

get_remote_json(*uri*, ***kwargs*)

class cove.lib.common.**CustomRefResolver**(**args*, ***kw*)
This RefResolver is only for use with the jsonschema library

resolve_remote(*uri*)

class cove.lib.common.**SchemaJsonMixin**

deref_schema(*schema_str*)

get_release_pkg_schema_fields()

get_release_pkg_schema_obj(*deref=False*)

get_release_schema_obj(*deref=False*)

release_pkg_schema_str

release_schema_str

cove.lib.common.**add_is_codelist**(*obj*)
This is needed so that we can detect enums that are arrays as the jsonschema library does not give you any parent information and the codelist property is on the parent in this case. Only applies to release.tag in core schema at the moment.

cove.lib.common.**common_checks_context**(*upload_dir*, *json_data*, *schema_obj*, *schema_name*, *context*, *extra_checkers=None*, *fields_regex=False*, *api=False*, *cache=True*)

cove.lib.common.**fields_present_generator**(*json_data*, *prefix=''*)

cove.lib.common.**get_additional_codelist_values**(*schema_obj*, *codelist_url*, *json_data*)

cove.lib.common.**get_counts_additional_fields**(*json_data*, *schema_obj*, *schema_name*, *context*, *fields_regex=False*)

cove.lib.common.**get_fields_present**(**args*, ***kwargs*)

cove.lib.common.**get_json_data_deprecated_fields**(*json_data_paths*, *schema_obj*)

cove.lib.common.**get_json_data_generic_paths**(*json_data*, *path=()*, *generic_paths=None*)

Transform json data into a dictionary with keys made of json paths.

Key are json paths (as tuples). Values are dictionaries with keys including specific indexes (which are not including in the top level keys), eg:

{'a': 'I am', 'b': ['a', 'list'], 'c': [{'ca': 'ca1'}, {'ca': 'ca2'}, {'cb': 'cb'}]}

will return:

generic_paths = { ('a',): {('a',): 'I am'}, ('b',): {

('b',): ['a', 'list'], ('b', 0): 'a', ('b', 1): 'list'

}, ('c',): {

('c',): [{'ca': 'ca1'}, {'ca': 'ca2'}, {'cb': 'cb'}

], ('c', 0): {'ca': 'ca1'}, ('c', 1): {'ca': 'ca2'}, ('c', 2): {'cb': 'cb'}

}, ('c', 'ca'): {

('c', 0, 'ca'): 'ca1', ('c', 1, 'ca'): 'ca2'

}, ('c', 'cb'): {('c', 2, 'cb'): 'cb'}

}

cove.lib.common.**get_json_data_missing_ids**(*json_data_paths*, *schema_obj*)

cove.lib.common.**get_orgids_prefixes**(*orgids_url=None*)

Get org-ids.json file from file system (or fetch upstream if it doesn't exist)

A lock file is needed to avoid different processes trying to access the file trampling each other. If a process has the exclusive lock, a different process will wait until it is released.

cove.lib.common.**get_schema_validation_errors**(*json_data*, *schema_obj*, *schema_name*, *cell_src_map*, *heading_src_map*, *extra_checkers=None*)

cove.lib.common.**get_spreadsheet_meta_data**(*upload_dir*, *file_name*, *schema*, *file_type='xlsx'*, *name='Meta'*)

cove.lib.common.**oneOf_draft4**(*validator*, *oneOf*, *instance*, *schema*)

oneOf_draft4 validator from https://github.com/Julian/jsonschema/blob/d16713a4296663f3d62c50b9f9a2893cb380b7af/jsonschema/_validators.py#L337 patched to sort the instance.

cove.lib.common.**required_draft4**(*validator*, *required*, *instance*, *schema*)

cove.lib.common.**schema_dict_fields_generator**(*schema_dict*)

cove.lib.common.**unique_ids**(*validator*, *ui*, *instance*, *schema*)

3.2 converters

3.3 exceptions

exception cove.lib.exceptions.**CoveInputDataError**(*context=None*)
An error that we think is due to the data input by the user, rather than a bug in the application.

exception cove.lib.exceptions.**UnrecognisedFileType**(*context=None*)

context

exception cove.lib.exceptions.**UnrecognisedFileTypeXML**(*context=None*)

context

cove.lib.exceptions.**cove_spreadsheet_conversion_error**(*func*)

cove.lib.exceptions.**cove_web_input_error**(*func*)

3.4 ocds

3.5 threesixtygiving

3.6 tools

class cove.lib.tools.**NumberStr**(*o*)

cove.lib.tools.**cached_get_request**

cove.lib.tools.**datetime_or_date**(*instance*)

cove.lib.tools.**decimal_default**(*o*)

cove.lib.tools.**get_file_type**(*file*)

cove.lib.tools.**get_no_exception**(*item*, *key*, *fallback*)

cove.lib.tools.**ignore_errors**(*f*)

cove.lib.tools.**to_list**(*item*)

cove.lib.tools.**update_docs**(*document_parent*, *counter*)

CHAPTER 4

Spreadsheet and CSV input.

Cove allows XLSX and CSV import in all the standards it supports. It uses the flattentool library to do the conversion from these formats to either XML or JSON. Please look at flattentool documentation for detailed information on how to create spreadsheet templates for a particular standard.

4.1 Metatab

Cove configures flattentool to allow an extra sheet in your spreadsheets (not for CSV) named "Meta" (case sensitive). This sheet contains items at the top level of your document. For JSON this means key/value pairs that appear at the top level object and in XML attributes on the outermost tag.

The "Meta" sheet is expected to be vertically aligned, so headings are on first column (not first row), and values are on second column. So a sheet named Meta could look like:

dataLicense	CC
version	2
publishedDate	2001-01-01

This will create a JSON object like:

```
{"dataLicense": "CC",
 "version: "2",
 "publishedDate": "2001-01-01",
 "someNestedData: [...]}
```

For XML like:

```
<toptag dataLicense="CC" version="2" publishedData="2001-01-01">
  ...
</toptag>
```

4.2 Hash command line at top of file.

For both CSV and Spreadsheet (XLSX) flattentool allows a special line at the top of the file. This line has to start with a "#" character in the first cell (i.e A1 in a spreadsheet) and nothing else. The rest of line contains commands to customize how the spreadsheet is parsed. For example:

#	skipRows 1	headerRows 2
this line	is	ignored
Some	Headings	Here
Some More	Headings	Here
some	data	here

Important: If there exists a hash command line at the top of the metatab sheet this will apply default commands across all other sheets (not the metatab itself). This can be overridden by supplying a hash commend line for a particular sheet.

The commands that flattentool (and therefore cove) allows are the following:

4.2.1 skipRows

This is followed by a number i.e `skipRows 3` and says how many rows at the top of the file (ignoring the hash line) will be ignored. For example:

#	skipRows 1	
this line	is	ignored
Some	Headings	Here
some	data	here

Defaults to 0 rows skipped.

4.2.2 headerRows

This is followed by a number i.e `headerRows 2` and says how many rows are header lines in the file. All header rows apart from the first one will be ignored. For example:

#	headerRows 2	
Some	Headings	Here
More	headings	here
some	data	here

Defaults to 1 header rows. 0 rows is invalid as cove needs a heading row.

4.2.3 ignore

This says that this whole sheet should not be looked at by cove:

#	ignore	
Everthing	ignored	on
this	sheet	

4.2.4 hashComments

This says that columns can be commented out by placing a # before the column name. If this command is used in the metatab it means that sheet names can be ignored by adding a # before the sheet name:

#	hashcomments	
Heading	# Ignored	Heading 2
some	ignored data	data

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index