README.rst

pyxform v1.12.0

pyxform is a Python library that makes writing forms for ODK Collect and Enketo easy by converting XLSForms (Excel spreadsheets) into ODK XForms. The XLSForms format is used in a number of tools.

XLS(X) documents used as input must follow to the XLSForm standard and the resulting output follows the ODK XForms standard.

pyxform is a major rewrite of xls2xform.

Running the latest release of pyxform

For those who want to convert forms at the command line, the latest official release of pyxform can be installed using pip:

```
pip install pyxform
```

The `xls2xform` command can then be used:

```
xls2xform path_to_XLSForm [output_path]
```

The currently supported Python versions for `pyxform` are 3.7, 3.8 and 3.9.

Running pyxform from local source

Note that you must uninstall any globally installed `pyxform` instance in order to use local modules. Please install java 8 or newer version.

From the command line, complete the following. These steps use a virtualenv to make dependency management easier, and to keep the global site-packages directory clean:

```
# Get a copy of the repository.
mkdir -P ~/repos/pyxform
cd ~/repos/pyxform
git clone https://github.com/XLSForm/pyxform.git repo
```

```
# Create and activate a virtual environment for the install.
/usr/local/bin/python3.8 -m venv venv
. venv/bin/activate

# Install the pyxform and it's production dependencies.
(venv)$ cd repo
(venv)$ pip install -e .
(venv)$ python pyxform/xls2xform.py --help
(venv)$ xls2xform --help           # same effect as previous line
(venv)$ which xls2xform            # ~/repos/pyxform/venv/bin/xls2xform
```

To leave and return to the virtualenv:

```
(venv)$ deactivate                 # leave the venv, scripts not on $PATH
$ xls2xform --help
# -bash: xls2xform: command not found
$ . ~/repos/pyxform/venv/bin/activate     # reactivate the venv
(venv)$ which xls2xform                   # scripts available on $PATH again
~/repos/pyxform/venv/bin/xls2xform
```

Installing pyxform from remote source

`pip` can install from the GitHub repository. Only do this if you want to install from the master branch, which is likely to have pre-release code. To install the latest release, see above.:

```
pip install git+https://github.com/XLSForm/pyxform.git@master#egg=pyxform
```

You can then run xls2xform from the commandline:

```
xls2xform path_to_XLSForm [output_path]
```

Development

To set up for development / contributing, first complete the above steps for "Running pyxform from local source", then complete the below.:

```
pip install -r dev_requirements.pip
```

You can run tests with:

```
nosetests
```

On Windows, use:

```
nosetests -v -v --traverse-namespace ./tests
```

Before committing, make sure to format the code using `black`:

```
black pyxform tests
```

If you are using a copy of black outside your virtualenv, make sure it is the same version as listed in requirements_dev.pip.

In case the pre-commit.sh hooks don't run, also run the code through `isort` (sorts imports) and `flake8` (misc code quality suggestions). The syntax is the same as above for `black`.

Writing tests

Make sure to include tests for the changes you're working on. When writing new tests you should add them in `tests` folder. Add to an existing test module, or create a new test module. Test modules are named after the corresponding source file, or if the tests concern many files then module name is the topic or feature under test.

When creating new test cases, where possible use `PyxformTestCase` as a base class instead of `unittest.TestCase`. The `PyxformTestCase` is a toolkit for writing XLSForms as MarkDown tables, compiling example XLSForms, and making assertions on the resulting XForm. This makes code review much easier by putting the XLSForm content inline with the test, instead of in a separate file. A `unittest.TestCase` may be used if the new tests do not involve compiling an XLSForm (but most will). Do not add new tests using the old style `XFormTestCase`.

When writing new `PyxformTestCase` tests that make content assertions, it is strongly recommended that the `xml__xpath*` matchers are used, in particular `xml__xpath_match`. Most older tests use matchers like `xml__contains` and `xml__excludes`, which are simple string matches of XML snippets against the result XForm. The `xml__xpath_match` kwarg accepts an XPath expression and expects 1 match. The main benefits of using XPath are 1) it allows specifying a document location, and 2) it does not require a particular document order for elements or attributes or whitespace output. To take full advantage of 1), the XPath expressions should specify the full document path (e.g. `/h:html/h:head/x:model`) rather than a search (e.g. `.//x:model`). To take full advantage of 2), the expression should include element predicates that specify the expected attribute values, e.g. `/h:html/h:body/x:input[@ref='/trigger-column/a']`. To specify the absence of an element, an expression like the following may be used with `xml__xpath_match`: `/h:html[not(descendant::x:input)]`, or alternatively `xml__xpath_count`: `.//x:input` with an expected count of 0 (zero).

Documentation

To check out the documentation for pyxform do the following:

```
pip install Sphinx==1.0.7
cd your-virtual-env-dir/src/pyxform/docs
make html
```

Change Log

Changelog

Releasing pyxform

1. Make sure the version of ODK Validate in the repo is up-to-date:

   ```
   pyxform_validator_update odk update ODK-Validate-vx.x.x.jar
   ```

2. Run all tests through Validate by setting the default for `run_odk_validate` to `kwargs.get("run_odk_validate", True)` in `tests/pyxform_test_case.py`.
3. Draft a new GitHub release with the list of merged PRs. Follow the title and description pattern of the previous release.
4. Checkout a release branch from latest upstream master.
5. Update `CHANGES.txt` with the text of the draft release.
6. Update `README.rst`, `setup.py`, `pyxform/__init__.py` with the new release version number.
7. Commit, push the branch, and initiate a pull request. Wait for tests to pass, then merge the PR.
8. Tag the release and it will automatically be published

Manually releasing

Releases are now automatic. These instructions are provided for forks or for a future change in process.

1. In a clean new release only directory, check out master.
2. Create a new virtualenv in this directory to ensure a clean Python environment:

```
/usr/local/bin/python3.8 -m venv pyxform-release
. pyxform-release/bin/activate
```

3. Install the production and packaging requirements:

```
pip install -e .
pip install wheel twine
```

4. Clean up build and dist folders:

```
rm -rf build dist pyxform.egg-info
```

5. Prepare `sdist` and `bdist_wheel` distributions:

```
python setup.py sdist bdist_wheel
```

6. Publish release to PyPI with `twine`:

```
twine upload dist/pyxform-*-py3-none-any.whl dist/pyxform-*.tar.gz
```

7. Tag the GitHub release and publish it.

Releases 38

v1.12.0 Latest
Jan 20, 2023

+ 37 releases

Packages

No packages published

Contributors 51

+ 40 contributors

Languages

- **Python** 100.0%