Spreadsheet Forms

Jan 04, 2023

Contents

Spreadsheet Forms is a Python library for working with data in JSON format from forms contained in spreadsheets.

It includes functions to:

- Create a blank spreadsheet form for collecting data
- Populate a spreadsheet form with existing data for review
- Extract JSON data from a spreadsheet form for storage and processing

The above functions require a guide form spreadsheet that specifies the structure of the form.

There can be several functions to achieve the same thing, based on what level of performance you need.

CHAPTER 1

Examples

1.1 A Guide Form spreadsheet

A guide form spreadsheet is a template that specifies the structure of a spreadsheet form. All functions require a guide form.

Guide forms use special values in certain cells to specify the structure of a spreadsheet form.

For example:

Pet	SPREADSHEETFORM:SINGLE:pet
Toys:	
Title	**Does it squeak?**
SPREADSHEETFORM:DOWN:likes/toys:title	SPREADSHEETFORM:DOWN:likes/toys:squeak

1.2 Extracting data from a spreadsheet

Given the guide form above and the following populated spreadsheet form:

Pet	Dog
Toys:	
Title	**Does it squeak?**
Plastic bone	Oh Yes
Tennis Ball	No

The function *get_data_from_form* will produce the following data:

```
{
    "pet": "Dog",
    "likes": {
        "toys": [
            {"title": "Plastic bone", "squeak": "Oh Yes"},
            {"title": "Tennis Ball", "squeak": "No"}
        ]
    }
}
```

Note the SINGLE keyword is turned into a field, but the DOWN row is turned into a list. The people filling in the spreadsheet can add as many or as few items to the DOWN table as they want.

1.3 Populating a spreadsheet form

The process can be run in reverse using the *put_data_in_form* function.

Given the JSON data above, the function will produce the populated spreadsheet form above.

CHAPTER 2

Requirements

Python 3.6+

Currently only works with Excel 2010+ xlsx files.

CHAPTER 3

Python API

3.1 make_empty_form function

3.1.1 Purpose

Generates a blank spreadsheet form based on the structure specified in a guide form.

3.1.2 Call

```
from spreadsheetforms.api import make_empty_form

make_empty_form(guide_filename, out_filename):
```

3.1.3 Inputs

Pass:

- guide_filename - filename of the guide spreadsheet
- out_filename - filename of the output spreadsheet

3.1.4 Outputs

Returns nothing; simply creates or replaces the out_filename file.

3.2 get_data_from_form function

3.2.1 Purpose

Extracts JSON data from a spreadsheet form, based on a structure specified in a guide form.

If performance is an issue, see *get_data_from_form_with_guide_spec*.

3.2.2 Call

```
from spreadsheetforms.api import get_data_from_form,␣
↪GetDataFromFormMissingWorksheetAction

data = get_data_from_form(
    guide_filename,
    in_filename,
    date_format=None,
    missing_worksheet_action=GetDataFromFormMissingWorksheetAction.RAISE_EXCEPTION
)
```

3.2.3 Inputs

Pass:

- *guide_filename* - filename of the guide spreadsheet
- *in_filename* - filename of the input spreadsheet
- *date_format* - if None, any date formatted cells in the spreadsheet will be returned as Python datetime.datetime objects. If set, they will be turned into strings using strftime. For format options, see Python docs .
- *missing_worksheet_action* - what to do if the guide spreadsheet specifies a worksheet that does not exist in the input spreadsheet.

Possible options for *missing_worksheet_action* are:

- *GetDataFromFormMissingWorksheetAction.RAISE_EXCEPTION* - raise an exception of class *spreadsheetforms.exceptions.MissingWorksheetException*
- *GetDataFromFormMissingWorksheetAction.SET_NO_DATA* - silently ignores the problem. The data keys that should have been set from the missing worksheet will just not exist in the output.

3.2.4 Outputs

Returns a JSON representation of the data extracted from the form.

3.3 get_guide_spec function

3.3.1 Purpose

Extracts a JSON representation of the structure of a spreadsheet form specified in a guide form.

You can use this in combination with *get_data_from_form_with_guide_spec* if *get_data_from_form* is to slow.

3.3.2 Call

```
from spreadsheetforms.api import get_guide_spec

data = get_guide_spec(guide_filename):
```

3.3.3 Inputs

Pass:

- guide_filename - filename of the guide spreadsheet

3.3.4 Outputs

Returns a JSON representation of the structure specified in the guide form.

3.4 get_data_from_form_with_guide_spec function

3.4.1 Purpose

Extracts JSON data from a spreadsheet form, based on a structure specified in JSON.

This is a version of *get_data_from_form* that should be used where performance is an issue.

get_data_from_form will parse the guide spreadsheet every time it is called. If called multiple times, or you need faster action when it is called, use this instead. Call *get_guide_spec* and cache the results. Pass that cached value to this function.

If performance is not an issue, we recommend just using *get_data_from_form* as that is simpler.

3.4.2 Call

```
from spreadsheetforms.api import get_data_from_form_with_guide_spec,␣
↪GetDataFromFormMissingWorksheetAction

data = get_data_from_form_with_guide_spec(
    guide_spec,
    in_filename,
    date_format=None,
    missing_worksheet_action=GetDataFromFormMissingWorksheetAction.RAISE_EXCEPTION
)
```

3.4.3 Inputs

Pass:

- guide_spec - Data from calling the *get_guide_spec* function

Other parameters are the same as specified for *get_data_from_form*.

3.4.4 Outputs

Returns a JSON block of the data it managed to extract from the input.

3.5 put_data_in_form function

3.5.1 Purpose

Populates a spreadsheet form with existing JSON data, based on the structure of a guide form.

3.5.2 Call

```
from spreadsheetforms.api import put_data_in_form

put_data_in_form(guide_filename, data, out_filename)
```

3.5.3 Inputs

Pass:

- guide_filename - filename of the guide spreadsheet
- data - a JSON block of the data
- out_filename - filename of the output spreadsheet

3.5.4 Outputs

Returns nothing; simply creates or replaces the out_filename file.

CHAPTER 4

Guide Form

A Guide form is a spreadsheet you create to act as a configuration for the forms you will work with.

You then put special values in certain cells to indicate that these cells should have data read from them or inserted into.

4.1 Single field

4.1.1 Definition

Sometimes you want a single cell in the spreadsheet to be represented by a single JSON value in the data.

To do this, in your cell put

```
SPREADSHEETFORM:SINGLE:jsonkey
```

See *JSON Key for information on how to structure those*.

4.1.2 Example

A guide of:

Pet	SPREADSHEETFORM:SINGLE:pet

And a spreadsheet of:

Pet	Cat

Will map to the data:

```
{
    "pet": "Cat"
}
```

4.2 Down field

4.2.1 Definition

Sometimes you want to allow people to put in multiple rows with the same set of headings. People can put in as few or as many rows as they want. Each row will be converted to one JSON dictionary. In the final data, there will be a list of JSON dictionaries.

To do this, in your cell put

```
SPREADSHEETFORM:DOWN:listkey:itemkey
```

The *listkey* is the JSON key that the list of items will appear in.

All Down configurations for the same *listkey* should appear on the same row.

You should probably only have one set of down configurations per sheet (ie only one *listkey* per sheet) and underneath that there should be nothing. This is because the user can put as many data rows as they want in; if you try and put something else there it may be overwritten.

The order of the rows and the order of the items in the JSON list will be the same.

The *itemkey* is the JSON key that the data will appear in in each dictionary.

See *JSON Key for information on how to structure those*.

4.2.2 Example

A guide of:

Title	Does it squeak?
SPREADSHEETFORM:DOWN:toys:title	SPREADSHEETFORM:DOWN:toys:squeak

And a spreadsheet of:

Title	Does it squeak?
Plastic bone	Oh Yes
Tennis Ball	No

Will map to the data:

```
{
    "toys": [
        {"title": "Plastic bone", "squeak": "Oh Yes"},
        {"title": "Tennis Ball", "squeak": "No"},
    ]
}
```

4.3 Right field

4.3.1 Definition

Sometimes you want to allow people to put in multiple columns with the same set of headings in rows. People can put in as few or as many columns as they want. Each column will be converted to one JSON dictionary. In the final data, there will be a list of JSON dictionaries.

To do this, in your cell put

```
SPREADSHEETFORM:RIGHT:listkey:itemkey
```

The *listkey* is the JSON key that the list of items will appear in.

All Down configurations for the same *listkey* should appear on the same row.

You should probably only have one set of right configurations per set of rows and to the right of that there should be nothing. This is because the user can put as many data columns as they want in; if you try and put something else there it may be overwritten.

The order of the columns and the order of the items in the JSON list will be the same.

The *itemkey* is the JSON key that the data will appear in in each dictionary.

See *JSON Key for information on how to structure those*.

4.3.2 Example

A guide of:

Title	SPREADSHEETFORM:RIGHT:toys:title
Does it squeak?	SPREADSHEETFORM:RIGHT:toys:squeak

And a spreadsheet of:

Title	Plastic bone	Tennis Ball
Does it squeak?	Oh Yes	No

Will map to the data:

```
{
    "toys": [
        {"title": "Plastic bone", "squeak": "Oh Yes"},
        {"title": "Tennis Ball", "squeak": "No"},
    ]
}
```

4.4 JSON Keys

In the guideform, you can set JSON Keys at various points.

These set the point in the data that values will be read or written to.

These can be single keys:

Eg:

- *pet*

These can also be several keys split by a slash. In this case, it will traverse down several dictionaries.

Eg:

- *pet/kind*