Spreadsheet Forms

Nov 06, 2020

Contents

You want to create a form in a spreadsheet. You want to send this to people who will fill in the data. You want to then get the forms back and automatically extract the data from them for you to process. Also, maybe you want to take some existing data and put it into a spreadsheet form to send to someone to check.

CHAPTER 1

Examples

A guide form spreadsheet is the template to show us where to expect data:

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

A spreadsheet of:

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

Will map to the data (and vice versa):

```
{
    "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. People can add as many or as few items to the DOWN table as they want.

CHAPTER 2

Requirements

Python 3.6+

Currently only works with Excel 2010+ xlsx files.

CHAPTER 3

Python API

3.1 make_empty_form function

```
make_empty_form(guide_filename, out_filename):
```

Pass:

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

Returns nothing; simply creates or replaces the out_filename file.

3.2 get_data_from_form function

```
data = get_data_from_form(
    guide_filename,
    in_filename,
    date_format=None
)
```

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 .

Returns:

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

3.3 get_guide_spec function

This takes a guide spreadsheet and returns the specification of the spreadsheet as a JSON block.

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

```
get_guide_spec(guide_filename):
```

Pass:

- guide_filename - filename of the guide spreadsheet

Returns:

- data - a JSON block of the specification for the guide form

3.4 get_data_from_form_with_guide_spec function

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.

```
data = get_data_from_form_with_guide_spec(
    guide_spec,
    in_filename,
    date_format=None
)
```

Pass:

- guide_spec - Data from calling the *get_guide_spec* function
- 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 .

Returns:

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

3.5 put_data_in_form function

```
put_data_in_form(guide_filename, data, out_filename)
```

Pass:

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

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*