Inetsix Lib Documentation

Release 0.3

Thomas Grimonet

Mar 26, 2019

Contents:

CHAPTER 1

Tools Module

class inetsix.tools.**InetsixTools**

Class to expose some useful functions for day to day scripting.

InetsixTools class provides a list of functions used in different script. Using such class to centralize tools help to maintain and enhance code instead of updating all scripts to be enhanced.

static dict_to_list(*dict_input=None*, *key=None*)

Transform a dictionary to a list.

Parameters

- **dict_input** (*dict()*) – Dictionary used to create list
- **key** (*str*) – If defined, only value from this key is extracted

Returns A list of entries

Return type list()

static download_file(*remote_file=None*, *verbose=False*)

Function to download file from HTTP server.

Download file from an HTTP or HTTPS server and save file locally

Parameters

- **remote_file** (*str*) – URL to download
- **verbose** – Manage verbosity level

Returns File name to access to local copy of the file

Return type str

static file_basename(*file_path*)

Extract Filename from complete path.

Parameters **file_path** (*str*) – file path to a file

Returns filename without path

Return type str

static get_list_files(*path=None*, *extension='j2'*, *debug=False*)
List files in a specific directory

Return a list of files found in a directory. A filter can be used to search specific file extensions.

Parameters

- **path** (*str*) – Path to search for file
- **extension** (*str*) – Extension to search files. Default is `j2`
- **debug** (*bool*) – Trigger to enable verbosity

Returns A list of files found in `path`

Return type list

static jprint(*data=[]*)
PPrint wrapper to display list and dict.

Parameters **data** (*list or dict*) – data to display python structure

static list_unique_values(*myList=[]*)
Remove duplicate entries in a list.

Parameters **myList** (*list()*) – List where duplicates must be removed

Returns List without duplicates

Return type list()

static load_constant(*key_name*, *default='UNSET'*, *verbose=False*)
Set up constant value from OS Environment.

Help to define CONSTANT from OS Environment. If it is not defined, then, fallback to default value provided within parameters

Example

```
>>> from inetsix.tools import InetsixTools
>>> CVP_USER = InetsixTools.load_constant(key_name='CVP_USER', default=
↪'username')
>>> print CVP_USER
myUsername
```

Parameters

- **key_name** (*str*) – VAR to lookup in os.environment
- **default** (*str*) – Default value to use if key_name is not defined.
- **verbose** (*bool*) – Boolean to activate verbos mode (Optional, expected values: debug level)

Returns Value for variable

Return type str

CHAPTER 2

Excel Module

2.1 Public Methods

Python class to serialize Excel spreadsheet to Python structures

Allow to extract data from an Excel sheet to convert it to different This class supports import from different tab from Excel file and can serialize both: table and list

class inetsix.excel.**ExcelSerializer**(*excel_path=None*, *sheet='Sheet1'*, *has_header=True*, *nb_columns=0*)

Convert Excel data into python structures.

Allow to extract data from an Excel sheet to convert it to different This class supports import from different tab from Excel file and can serialize both: table and list

Example Output example for a table

```
>>> table = ExcelSerializer(excel_path='../excel/input.xlsx')
>>> table.serialize_table(sheet="Topology", nb_columns=6)
>>> print table.get_data(sheet="Topology")
[   {   'id': '1',
    u'local_device': u'dev1',
    u'local_port': u'port1',
    u'local_port_name': u'et-0/0/0',
    u'remote_device': u'dev2',
    u'remote_port': u'port1',
    u'remote_port_name': u'et-0/0/0'},
{   'id': '2',
    u'local_device': u'dev1',
    u'local_port': u'port2',
    u'local_port_name': u'et-0/0/1',
    u'remote_device': u'dev3',
    u'remote_port': u'port1',
    u'remote_port_name': u'et-0/0/1'},
{   'id': '3',
```

(continues on next page)

(continued from previous page)

```
    u'local_device': u'dev2',
    u'local_port': u'port3',
    u'local_port_name': u'et-0/0/2',
    u'remote_device': u'dev4',
    u'remote_port': u'port1',
    u'remote_port_name': u'et-0/0/2'}]
```

Example Output example for a list

```
>>> table = ExcelSerializer(excel_path='../excel/input.xlsx')
>>> table.serialized_list(sheet="List")
>>> print table.get_data(sheet="List")
{   'asn_base': '65000',
    'bgp_export': 'underlay-export',
    'bgp_group': 'underlay',
    'bgp_import': 'underlay-import',
    'mtu_phy_int': ['9200', '1000']
}
```

Todo: Create a method to locate data in Excel Sheet.

_ExcelSerializer__find_header (*sheet=None*, *nb_columns=2*)

Look for array in the table and provide list of them.

Extract table header and create a dict() to store them with a standardize name: lower and replace space by "_"

Parameters

- **sheet** (*str.*) – Sheet name to read (default=None, optional))
- **nb_columns** (*int.*) – Number of column to read (optional)

Returns list of headers found in given sheet

Return type list

_serialize_cell (*xlsCellStr=None*)

Serialize a cell by doing some cleanup.

Transform multi lines content to a single line str.

Parameters **xlsCellStr** (*str*) – content of a cell to serialize and cleanup (default=None)

Returns Single line string

Return type str

_string_cleanup (*string*)

Cleanup a string to lower and remove space.

Parameters **string** (*str*) – String to cleanup

Returns Cleanup string

Return type str

get_data (*sheet=None*)

Provide a read only access to table data

Note: This function is a wrapper to `get_yaml()` function

Example

```
>>> table = ExcelSerializer(excel_path='../excel/input.xlsx')
>>> table.serialized_list(sheet="List")
>>> print table.get_data(sheet="List")
{   'asn_base': '65000',
    'bgp_export': 'underlay-export',
    'bgp_group': 'underlay',
    'bgp_import': 'underlay-import',
    'mtu_phy_int': ['9200', '1000']
}
```

Parameters **sheet** (*str.*) – Sheet name to read (default=None, optional))

Returns A YAML structure with data found in given sheet

Return type dict

get_header()

Provide a read only access to header table data.

Example

```
>>> table = ExcelSerializer(excel_path='../excel/input.xlsx')
>>> table.serialize_table(sheet="Topology", nb_columns=6)
>>> print table.get_header()
[   'asn_base',
    'bgp_export',
    'bgp_group',
    'bgp_import',
    'mtu_phy_int'
]
```

Returns A YAML structure with headers found in given sheet

Return type list

get_yaml(*sheet=None*)

Return a per sheet Python structure compatible with YAML language

Example

```
>>> table = ExcelSerializer(excel_path='../excel/input.xlsx')
>>> table.serialized_list(sheet="List")
>>> print table.get_yaml(sheet="List")
{   'asn_base': '65000',
    'bgp_export': 'underlay-export',
    'bgp_group': 'underlay',
    'bgp_import': 'underlay-import',
    'mtu_phy_int': ['9200', '1000']
}
```

Parameters **sheet** (*str.*) – Sheet name to read (default=None, optional))

Returns A YAML structure with data found in given sheet

Return type dict

get_yaml_all()
Get Yaml data.

Provide a dictionary of all data found in all sheets parsed in this Excel file.

Example

```
>>> table = ExcelSerializer(excel_path='../excel/input.xlsx')
>>> table.serialized_list(sheet="List")
>>> table.serialize_table(sheet="Topology", nb_columns=6)
>>> table.get_yaml_all()
{   'List': {
        'asn_base': '65000',
        'bgp_export': 'underlay-export',
        'bgp_group': 'underlay',
        'bgp_import': 'underlay-import',
        'mtu_phy_int': ['9200', '1000']
    },
    'Topology': [
       {   'id': '1',
            u'local_device': u'dev1',
            u'local_port': u'port1',
            u'local_port_name': u'et-0/0/0',
            u'remote_device': u'dev2',
            u'remote_port': u'port1',
            u'remote_port_name': u'et-0/0/0'},
        {   'id': '2',
            u'local_device': u'dev1',
            u'local_port': u'port2',
            u'local_port_name': u'et-0/0/1',
            u'remote_device': u'dev3',
            u'remote_port': u'port1',
            u'remote_port_name': u'et-0/0/1'},
        {   'id': '3',
            u'local_device': u'dev2',
            u'local_port': u'port3',
            u'local_port_name': u'et-0/0/2',
            u'remote_device': u'dev4',
            u'remote_port': u'port1',
            u'remote_port_name': u'et-0/0/2'}]
    ]
}
```

Parameters **sheet** (*str.*) – Sheet name to read (default=None, optional))

Returns A YAML structure with data found in sheets parsed previously

Return type dict

serialize_list(*sheet=None*, *nb_columns=2*)
Serialize a list into a YAML compliant format.

Example

```
>>> table = ExcelSerializer(excel_path='../excel/input.xlsx')
>>> table.serialized_list(sheet="List")
```

Note: Values are also saved in `self._data_array` for later use.

Parameters

- **sheet** (*str.*) – Sheet name to read (default=None, optional))
- **nb_columns** (*int.*) – Number of column to read (optional)

Returns YAML compatible dictionnary with all extracted values from the list

Return type dict

serialize_table(*sheet=None*, *nb_columns=2*, *has_header=True*)

Serialize Excel file into python structure.

Open Excel file defined in `self.__filename` and look for data Because columns are using letters instead of numbers, it is required to convert decimal values to alphabet letters.

For every line, function is reading every cell and move them to a `dict` entry Once line is completely read, `dict` is appended to a list of all lines.

Example

```
>>> table = ExcelSerializer(excel_path='../excel/input.xlsx')
>>> table.serialize_table(sheet="Topology", nb_columns=6)
```

Note: Values are also saved in `self._data_array` for later use.

Parameters

- **sheet** (*str.*) – Sheet name to read
- **nb_columns** (*int.*) – Number of column to read
- **has_header** (*bool.*) – Boolean to enable header finding.

Returns YAML compatible list of all extracted lines

Return type list

2.2 Private Methods

Python class to serialize Excel spreadsheet to Python structures

Allow to extract data from an Excel sheet to convert it to different This class supports import from different tab from Excel file and can serialize both: table and list

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

i

Index

Symbols

D

E

F

G

I

J

L

S