pyexcel Documentation

Release 0.0.1

C. W.

October 18, 2014

Contents

CHAPTER 1

Core packages

1.1 pyexcel.readers module

1.1.1 pyexcel.readers

Uniform interface for reading different excel file formats

copyright

3. 2014 by C. W.

license GPL v3

class pyexcel.readers.**CSVReader**(*file*)
csv reader

cell_value(*row*, *column*)
Random access to the csv cells

number_of_columns()
Number of columns in the csv file

assuming the length of each row is uniform

number_of_rows()
Number of rows in the csv file

class pyexcel.readers.**ColumnFilterableSeriesReader**(*file*)
Bases: pyexcel.readers.GenericSeriesReader

Columns can be filtered but not rows

filter(*filter*)

class pyexcel.readers.**FilterableReader**(*file*)
Bases: pyexcel.readers.Reader

Reader that can be applied one filter

cell_value(*row*, *column*)
Random access to the data cells

column_range()

filter(*afilter*)

number_of_columns()
Number of columns in the data file

number_of_rows()
Number of rows in the data file

row_range()

class pyexcel.readers.**GenericSeriesReader**(*reader*)
Bases: pyexcel.readers.FilterableReader

For data with column headers

x y z 1 2 3 4 5 6

This class has a default filter that filter out row 0 as headers. Extra functions were added to return headers at row 0

named_column_at(*name*)

series()

class pyexcel.readers.**ODSReaderImp**(*file*)
Bases: pyexcel.readers.CSVReader

ods reader

Currently only support first sheet in the file

class pyexcel.readers.**Reader**(*file*)
Wrapper class to unify csv, xls and xlsx reader

cell_value(*row*, *column*)
Random access to the data cells

column_at(*index*)
Returns an array that collects all data at the specified column

column_range()
Utility function to get column range

columns()
Returns a column iterator to go through each column from left to right

contains(*predicate*)

enumerate()
Default iterator to go through each cell one by one from top row to bottom row and from left to right

number_of_columns()
Number of columns in the data file

number_of_rows()
Number of rows in the data file

rcolumns()
Returns a column iterator to go through each column from right to left

reverse()
Reverse iterator to go through each cell one by one from bottom row to top row and from right to left

row_at(*index*)
Returns an array that collects all data at the specified row

row_range()
Utility function to get row range

rows()
Returns a row iterator to go through each row from top to bottom

rrows()
Returns a row iterator to go through each row from bottom to top

rvertical()
Default iterator to go through each cell one by one from rightmost column to leftmost row and from bottom to top

vertical()
Default iterator to go through each cell one by one from leftmost column to rightmost row and from top to bottom

class pyexcel.readers.**SeriesReader**(*file*)
Bases: pyexcel.readers.GenericSeriesReader

rows other than header row can be filtered. row number has been shifted by 1 as header row is protected.

columns can be filtered.

filter(*afilter*)

named_column_at(*name*)

series()

class pyexcel.readers.**StaticSeriesReader**(*file*)
Bases: pyexcel.readers.GenericSeriesReader

Static Series Reader. No filters can be applied.

class pyexcel.readers.**XLSReader**(*file*)
xls reader

Currently only support first sheet in the file

cell_value(*row*, *column*)
Random access to the xls cells

number_of_columns()
Number of columns in the xls file

number_of_rows()
Number of rows in the xls file

1.2 pyexcel.writers module

1.2.1 pyexcel.writers

Uniform interface for writing different excel file formats

copyright

3. 2014 by C. W.

license GPL v3

class pyexcel.writers.**CSVWriter**(*file*)
csv file writer

close()
This call close the file handle

write_row(*array*)
write a row into the file

class pyexcel.writers.**ODSWriter**(*file*)
open document spreadsheet writer

close()
This call writes file

write_row(*array*)
write a row into the file

class pyexcel.writers.**Writer**(*file*)
Uniform excel writer

It provides one interface for writing ods, csv, xls, xlsx and xlsm

close()
Close the writer

Please remember to call close function

write_array(*table*)
Write a table

table can be two dimensional array or a row iterator

write_dict(*the_dict*)
Write a whole dictionary

series and data will be write into one file

write_reader(*reader*)
Write a pyexcel reader

In this case, you may use FiterableReader or SeriesReader to do filtering first. Then pass it onto this function

write_row(*array*)
Write a row

write a row into the file in memory

class pyexcel.writers.**XLSWriter**(*file*)
xls, xlsx and xlsm writer

close()
This call actually save the file

write_row(*array*)
write a row into the file

1.3 pyexcel.iterators module

1.3.1 pyexcel.iterators

Iterate through the pyexcel readers

copyright

3. 2014 by C. W.

license GPL v3

class pyexcel.iterators.**ColumnIterator**(*reader*)
Column Iterator from left to right

next()

class pyexcel.iterators.**ColumnReverseIterator**(*reader*)
Column Reverse Iterator from right to left

next()

class pyexcel.iterators.**HBLTRIterator**(*reader*)
Bases: pyexcel.iterators.VBLTRIterator

Horizontal Bottom Left to Top Right Iterator

Iterate horizontally from bottom left to top right >>E >>> ^ S>> |

get_next_value()

class pyexcel.iterators.**HBRTLIterator**(*reader*)
Bases: pyexcel.iterators.HTLBRIterator

Iterate horizontally from bottom right to top left

exit_condition()

get_next_value()

move_cursor()

class pyexcel.iterators.**HTLBRIterator**(*reader*)
Iterate horizontally from top left to bottom right

default iterator for Reader class

exit_condition()
Determine if all data have been iterated

get_next_value()
Get next value

move_cursor()
move internal cursor

next()
determine next value

this function is further divided into small functions so that other kind of iterators can easily change its behavior

next_cell_position()
Determine next cell position

class pyexcel.iterators.**HTRBLIterator**(*reader*)
Horizontal Top Right to Bottom Left Iterator

Iterate horizontally from top right to bottom left <<S <<< E<<

exit_condition()

get_next_value()

next()

class pyexcel.iterators.**RowIterator**(*reader*)
Iterate data row by row from top to bottom

next()

class pyexcel.iterators.**RowReverseIterator**(*reader*)
Iterate data row by row from bottom to top

next()

class pyexcel.iterators.**SeriesColumnIterator**(*reader*)
Column Iterator

next()

class pyexcel.iterators.**VBLTRIterator**(*reader*)
Bases: pyexcel.iterators.HTRBLIterator

Vertical Bottom Left to Top Right Iterator

Iterate vertically from bottom left to top right ^^E ^^^ S^^ ->

exit_condition()

get_next_value()

class pyexcel.iterators.**VBRTLIterator**(*reader*)
Bases: pyexcel.iterators.HBRTLIterator

Iterate vertically from bottom right to top left

next_cell_position()

class pyexcel.iterators.**VTLBRIterator**(*reader*)
Bases: pyexcel.iterators.HTLBRIterator

Iterate vertically from top left to bottom right

next_cell_position()
this function controls the iterator's path

class pyexcel.iterators.**VTRBLIterator**(*reader*)
Bases: pyexcel.iterators.HTRBLIterator

Vertical Top Right to Bottom Left Iterator

Iterate horizontally from top left to bottom right ||S ||| E||

get_next_value()

1.4 pyexcel.filters module

Design note for filter algorithm

#1 2 3 4 5 6 7 <- original index # x x #1 3 4 6 7 <- filtered index #1 2 3 4 5 <- actual index after filtering

class pyexcel.filters.**ColumnFilter**(*indices*)
Bases: pyexcel.filters.ColumnIndexFilter

class pyexcel.filters.**ColumnIndexFilter**(*func*)

columns()

rows()

translate(*row*, *column*)

validate_filter(*reader*)

class pyexcel.filters.**EvenColumnFilter**
Bases: pyexcel.filters.ColumnIndexFilter

class pyexcel.filters.**EvenRowFilter**
Bases: pyexcel.filters.RowIndexFilter

Filter out even rows

row 0 is seen as the first row

class pyexcel.filters.**OddColumnFilter**
Bases: pyexcel.filters.ColumnIndexFilter

class pyexcel.filters.**OddRowFilter**
Bases: pyexcel.filters.RowIndexFilter

Filter out odd rows

row 0 is seen as the first row

class pyexcel.filters.**RowFilter**(*indices*)
Bases: pyexcel.filters.RowIndexFilter

class pyexcel.filters.**RowInFileFilter**(*reader*)
Bases: pyexcel.filters.RowValueFilter

class pyexcel.filters.**RowIndexFilter**(*func*)

columns()

rows()

translate(*row*, *column*)

validate_filter(*reader*)

class pyexcel.filters.**RowValueFilter**(*func*)
Bases: pyexcel.filters.RowIndexFilter

validate_filter(*reader*)

CHAPTER 2

Utility packages

2.1 pyexcel.cookbook package

2.1.1 pyexcel.cookbook

Cookbook for pyexcel

copyright

3. 2014 by C. W.

license GPL v3

pyexcel.cookbook.**merge_files**(*file_array*, *outfilename='pyexcel_merged.csv'*)
merge many files

pyexcel.cookbook.**merge_readers**(*reader_array*, *outfilename='pyexcel_merged.csv'*)
merge many readers

With FilterableReader and SeriesReader, you can do custom filtering

pyexcel.cookbook.**merge_two_files**(*file1*, *file2*, *outfilename='pyexcel_merged.csv'*)
merge two files

pyexcel.cookbook.**merge_two_readers**(*reader1*, *reader2*, *outfilename='pyexcel_merged.csv'*)
merge two readers

pyexcel.cookbook.**update_columns**(*infilename*, *column_dicts*, *outfilename=None*)
Update one or more columns of a data file with series

2.2 pyexcel.cookbook package

2.2.1 pyexcel.utils

Utility functions for pyexcel

copyright

3. 2014 by C. W.

license GPL v3

pyexcel.utils.**to_array**(*iterator*)
convert a reader iterator to an array

pyexcel.utils.**to_dict**(*iterator*)
convert a reader iterator to a dictionary

pyexcel.utils.**to_one_dimensional_array**(*iterator*)
convert a reader to one dimensional array

2.3 pyexcel.ext package

2.3.1 pyexcel.ext.odsreader

Uniform interface for writing different excel file formats

copyright

3. 2011 by Marco Conti

license Apache License 2.0

class pyexcel.ext.odsreader.**ODSReader**(*file*)

getSheet(*name*)

readSheet(*sheet*)

p

C

E

F

G

H

M

N

O

P

R

S

T

U

V

W

X