Flask-Excel

发布 *0.0.7*

C.W.

2021 年 07 月 10 日

Contents

作者 C.W.

源文件 http://github.com/pyexcel-webwares/Flask-Excel.git

提交问题 http://github.com/pyexcel-webwares/Flask-Excel/issues

发布的版本 0.0.7

文档生成日期 2021 年 07 月 10 日

以下是一段典型的开发人员和用户的对话:

```
   用户: "我上传了一个 excel 文件但是你的网页说文件格式不支持。"
开发人员: "哪你上传的是 xlsx 格式还是 csv 格式？"
   用户: "嗯，我不清楚。总之，我用的微软的 Excel 存的文件。哪一定是 excel 文件啦！"
开发人员: "嗨，事情是这样：从第一天开始，你就没有告诉我要支持所有的 excel 格式。"
          "要么，将就一下。要么，把项目推迟 N 天。"
```

Flask-Excel 是基于 pyexcel 软件库。它的使命是让大家在网站开发的时候，轻松的在数据存成 excel 文件让用户下载和处理用户上传的 excel 文件。它可以把 excel 数据转换成二维数组，一维的字典数组和以 excel 单页名为关键字，二维数组为值的字典；反之亦然。这样的话，任由文件格式变化，你都可以在以下三个场景自由做数据转换:

1. excel 文件上传和下载
2. 数据库输入输出
3. excel 数据分析和存储

同时，**Flask-Excel** 有以下两个保证:

1. 不管是任何文件格式，函数界面不变
2. 不管是数据存在哪里，函数界面不变

那么你就可以专注基于 Flask 的网站开发了。

最开始的时候，作者遇到一个可用性的问题：当一个简单的 excel 处理的网页交到用户手上的时候，这些用户，像行政助理，人力资源管理人员，老会抱怨网页不好用。事实上，并不是所有人都知道 csv, xls, xslx 各有什么区别。与其花时间教育用户相关的微软办公室软件的用法，不如把人类已知 excel 都给支持一下好了。同时为了在不修改代码情况下，我们能够通过装一个插件就不一个新的 excel 格式支持了，**pyexcel** 的编程界面做了很好的抽象处理。在整个 Python 社区，作者希望此软件包成为给 pandas 跑龙套的小包包。

可圈可点的性能：

1. 为 excel 数据导入数据库和从数据库输出数据为 excel 格式提供一站式服务
2. 把上传的 excel 文件直接转换成 Python 数据结构
3. 把 Python 数据结构转换为 excel 文件让用户下载
4. 在服务器里，把数据存成 excel 文件

5. 支持 csv, tsv, csvz, tsvz 格式。其他格式有以下软件包支持：

表 1: 文件格式插件列表

包包名字	文件格式	依赖
pyexcel-io	csv, csvz[1], tsv, tsvz[2]	
pyexcel-xls	xls, xlsx(read only), xlsm(read only)	xlrd, xlwt
pyexcel-xlsx	xlsx	openpyxl
pyexcel-ods3	ods	pyexcel-ezodf, lxml
pyexcel-ods	ods	odfpy

表 2: 专门读或写的插件

包包名字	文件格式	依赖
pyexcel-xlsxw	xlsx(write only)	XlsxWriter
pyexcel-libxlsxw	xlsx(write only)	libxlsxwriter
pyexcel-xlsxr	xlsx(read only)	lxml
pyexcel-xlsbr	xlsb(read only)	pyxlsb
pyexcel-odsr	read only for ods, fods	lxml
pyexcel-odsw	write only for ods	loxun
pyexcel-htmlr	html(read only)	lxml,html5lib
pyexcel-pdfr	pdf(read only)	camelot

[1] 压缩了的 csv 文件
[2] 压缩了的 tsv 文件

CHAPTER 1

插件使用指南

从 2020 年开始，所有 pyexcel-io 的插件都需要至少 python 3.6 了。如果需要支持以前的 python 版本，请继续使用 0.5.x 。

除了 csv 文件，xls, xlsx 和 ods 文件都是一个压缩文件。里面都是 xml 文件。

只有专门的读写插件可以边读边用或者边转换边写。

如果管理所有已经装上了的插件呢? 很简单，你可以用 pip 添加需要的插件，或者卸载不需要的插件。如果你有不同的项目，而且项目的依赖不一样，作者推荐用 python 的 venv 来给你的每一个项目创建一个新的虚拟 python 环境。有个别情况，两个插件需要共存，比如 pyexcel-ods 和 pyexcel-odsr，前者可以写 ods 文件，但你需要后者来读 ods 文件。在这种情形下呢，你可以用 library 变量，比如 get_array(‘my.ods’ , library=’ pyexcel-odsr’)。

表 1: 其他的格式

包包名字	文件格式	依赖	Python 版本
pyexcel-text	write only:rst, mediawiki, html, latex, grid, pipe, orgtbl, plain simple read only: ndjson r/w: json	tabulate	2.6, 2.7, 3.3, 3.4 3.5, 3.6, pypy
pyexcel-handsontable	handsontable in html	handsontable	same as above
pyexcel-pygal	svg chart	pygal	2.7, 3.3, 3.4, 3.5 3.6, pypy
pyexcel-sortable	sortable table in html	csvtotable	same as above
pyexcel-gantt	gantt chart in html	frappe-gantt	except pypy, same as above

CHAPTER 2

安装

你可以通过 pip 安装 Flask-Excel：

```
$ pip install Flask-Excel
```

或者复制到本地再安装：

```
$ git clone https://github.com/pyexcel-webwares/Flask-Excel.git
$ cd Flask-Excel
$ python setup.py install
```

每个插件的安装方法都有各自的文档。拿 xlsx 为例，你需要装 pyexcel-xlsx

```
$ pip install pyexcel-xlsx
```

CHAPTER 3

配置

在你的应用里，你必须加上以下的初始化代码：

```
import flask_excel as excel

...
excel.init_excel(app) # required since version 0.0.7
```

CHAPTER 4

快速上手

一个最简单的应用可以写这么短:

```
# -*- coding: utf-8 -*-
"""
tiny_example.py
:copyright: (c) 2015 by C. W.
:license: GPL v3 or BSD
"""
from flask import Flask, request, jsonify
import flask_excel as excel

app = Flask(__name__)


@app.route("/upload", methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        return jsonify({"result": request.get_array(field_name='file')})
    return '''
    <!doctype html>
    <title>Upload an excel file</title>
    <h1>Excel file upload (csv, tsv, csvz, tsvz only)</h1>
    <form action="" method=post enctype=multipart/form-data><p>
    <input type=file name=file><input type=submit value=Upload>
```

(下页继续)

(续上页)

```
    </form>
    '''


@app.route("/download", methods=['GET'])
def download_file():
    return excel.make_response_from_array([[1, 2], [3, 4]], "csv")


@app.route("/export", methods=['GET'])
def export_records():
    return excel.make_response_from_array([[1, 2], [3, 4]], "csv",
                                          file_name="export_data")


@app.route("/download_file_named_in_unicode", methods=['GET'])
def download_file_named_in_unicode():
    return excel.make_response_from_array([[1, 2], [3, 4]], "csv",
                                          file_name=u"中文文件名")


# insert database related code here
if __name__ == "__main__":
    excel.init_excel(app)
    app.run()
```

这个小应用有四个链接

1. 一个用来展示 Excel 文件上传。
2. 三个用来展示 Excle 文件下载。

第一个链接可以让你上传一个 Excel 文件，然后你会得到用 json 表示的文件内容。你可以用这个准备好的 样板文件 。当然你也可以用你自己的文件。在处理文件上传的代码里，我们用的是*get_array()* 。**get_array** 的参数 **file** 其实已经写在了网页里了:

```
<input ... name=file>
```

警告： 如果 **field_name** 没有用到的话，你的浏览器会给出”Bad Request: The browser (or proxy) sent a request that this server could not understand.”什么意思呢？正确的用法是：request.get_file(field_name=’file’)。错误的用法是：request.get_array(‘file’)。

其余的链接呢，只要你用浏览器访问，它们会简单的回复一个 cvs 文件，比如：http://localhost:50000/download/。

在这里，我们展示了 `make_response_from_array()` 如果把一个二维数组转换成你需要的 Excel 文件

CHAPTER 5

更多的文件格式

实例项目支持 csv, tsv 和他们的压缩版本: csvz and tsvz. 如果你需要其他的格式支持，请参照:ref:*file-format-list*，你可以装一个或所有的:

```
pip install pyexcel-xls
pip install pyexcel-xlsx
pip install pyexcel-ods
```

CHAPTER 6

数据库：数据输入和输出

继续前面的例子，我们来看看如果和数据库连起来。你可以把下面的代码拷贝到它们对应的地方，我们来一起做。

```
# insert database related code here
```

或者呢，你可以看这个已经完成了的 '数据库例子 <https://github.com/pyexcel/Flask-Excel/blob/master/examples/database_example.py>'_

现在我们先加入这些引入:

```
from flask_sqlalchemy import SQLAlchemy # sql operations
```

在你的命令行，运行下面的命令，装上 pyexcel-xls 和 pyexcel-handsontable:

```
pip install pyexcel-xls, pyexcel-handsontable
```

接着我们配置数据库链接。在这里我们用 Sqllite 。你在当前目录可以找到 **tmp.db**

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///tmp.db'
db = SQLAlchemy(app)
```

再拷贝数据库声明:

```
class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(80))
```

(下页继续)

(续上页)

```
    body = db.Column(db.Text)
    pub_date = db.Column(db.DateTime)

    category_id = db.Column(db.Integer, db.ForeignKey('category.id'))
    category = db.relationship('Category',
                               backref=db.backref('posts',
                                                  lazy='dynamic'))

    def __init__(self, title, body, category, pub_date=None):
        self.title = title
        self.body = body
        if pub_date is None:
            pub_date = datetime.utcnow()
        self.pub_date = pub_date
        self.category = category

    def __repr__(self):
        return '<Post %r>' % self.title


class Category(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(50))

    def __init__(self, name):
        self.name = name

    def __repr__(self):
        return '<Category %r>' % self.name
```

然后创建数据库模型:

```
db.create_all()
```

再写一个数据输入的函数:

```
@app.route("/import", methods=['GET', 'POST'])
def doimport():
    if request.method == 'POST':

        def category_init_func(row):
            c = Category(row['name'])
```

(下页继续)

(续上页)

```
            c.id = row['id']
            return c

        def post_init_func(row):
            c = Category.query.filter_by(name=row['category']).first()
            p = Post(row['title'], row['body'], c, row['pub_date'])
            return p
        request.save_book_to_database(
            field_name='file', session=db.session,
            tables=[Category, Post],
            initializers=[category_init_func, post_init_func])
        return redirect(url_for('.handson_table'), code=302)
    return '''
    <!doctype html>
    <title>Upload an excel file</title>
    <h1>Excel file upload (xls, xlsx, ods please)</h1>
    <form action="" method=post enctype=multipart/form-data><p>
    <input type=file name=file><input type=submit value=Upload>
    </form>
    '''
```

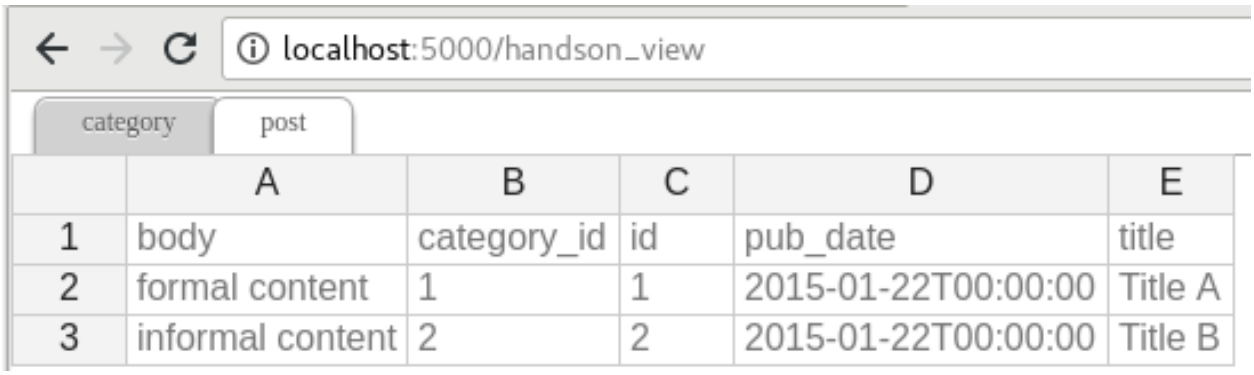
解释一下，*category_init_func* 和 *post_init_func* 是给 Category 和 Post 的自定义模型初始化函数。flask_excel 会把输入的 Excel 文件分批一行一行的把数据给初始化函数。一般初始化函数返回数据库模型的实例。初始化函数的返回值有个特殊用途：如果返回值是 None，哪所在的 Excel 文件的一行数据就会被丢掉。

我们再来写数据库输出的代码：

```
@app.route("/export", methods=['GET'])
def doexport():
    return excel.make_response_from_tables(db.session, [Category, Post], "xls")
```

再运行一下，打开 Visit http://localhost:5000/import 然后上传这个文件 sample-data.xls.

你会得到下面的网页：



	A	B	C	D	E
1	body	category_id	id	pub_date	title
2	formal content	1	1	2015-01-22T00:00:00	Title A
3	informal content	2	2	2015-01-22T00:00:00	Title B

以上的页面是由 pyexcel-handsontable 生成的。你需要做的就是用‘handsontable.html’文件扩展名:

```
@app.route("/handson_view", methods=['GET'])
def handson_table():
    return excel.make_response_from_tables(
        db.session, [Category, Post], 'handsontable.html')
```

Then visit http://localhost:5000/export to download the data back.

CHAPTER 7

输出过滤过的查询

前面的例子介绍了如果把一个或多个数据的表转换成 Excel 文件给用户下载。现在这个例子讲讲如何过滤一个表然后给用户下载。pass a query sets and an array of selected column names to *make_response_from_query_sets()* 允许你给一个查询和选中的栏目名字并给出一个单页的 Excel 文件下载：

```
@app.route("/custom_export", methods=['GET'])
def docustomexport():
    query_sets = Category.query.filter_by(id=1).all()
    column_names = ['id', 'name']
    return excel.make_response_from_query_sets(query_sets, column_names, "xls")
```

你可以打开这个链接看看：Then visit http://localhost:5000/custom_export .. _data-types-and-its-conversion-funcs:

CHAPTER 8

所有支持的数据结构

示例应用展示了数列，并不代表只有数列，其他的数据结构也是支持的。以下是所有的数据结构列表:

数据结构	从文件到数据结构	从数据结构到 http 回复
字典 (dict)	*get_dict()*	*make_response_from_dict()*
字典列表（records)	*get_records()*	*make_response_from_records()*
二维数组（a list of lists）	*get_array()*	*make_response_from_array()*
以二维数组为值的字典 (dict of a list of lists)	*get_book_dict()*	*make_response_from_book_dict()*
pyexcel.Sheet	*get_sheet()*	*make_response()*
pyexcel.Book	*get_book()*	*make_response()*
数据库表 (database table)	*save_to_database()* *isave_to_database()*	*make_response_from_a_table()*
一组数据库表 (a list of database tables)	*save_book_to_database()* *isave_book_to_database()*	*make_response_from_tables()*
数据库查询（a database query sets）		*make_response_from_query_sets()*
字典产生器（a generator for records）	iget_records()	
数组产生器（a generator of lists）	iget_array()	

需要更多信息的话，可以参照 pyexcel documentation

CHAPTER 9

支持

如果您觉得作者的付出对您有帮助，您可以给作者小女儿送个小玩具。谢谢您的支持!

CHAPTER 10

函数参考

Flask-Excel 把 **pyexcel** 函数安插到了 `flask.Request` 类里。

10.1 ExcelRequest

`flask_excel.ExcelRequest.`**`get_sheet`**(*field_name=None*, *sheet_name=None*, ***keywords*)

参数

- **`sheet_name`** –对于多个表单的 excel 文件, 它可以用来指定从哪一个表单取数据。缺省值是第一个表单。要是 csv , tsv 文件的话，可以忽略 *sheet_name* 。
- **`keywords`** –其他 `pyexcel.get_sheet()` 的参数

返回 `pyexcel.Sheet`

在下面的网页里，*field_name* 必须是“file”：

```
<!doctype html>
<title>Upload an excel file</title>
<h1>Excel file upload (csv, tsv, csvz, tsvz only)</h1>
<form action="" method=post enctype=multipart/form-data><p>
<input type=file name=file><input type=submit value=Upload>
</form>
```

flask_excel.ExcelRequest.**get_array**(*field_name=None*, *sheet_name=None*, ***keywords*)

参数

- **field_name** –和前面*get_sheet()* 一样。
- **sheet_name** –和前面*get_sheet()* 一样。
- **keywords** –其他 pyexcel.get_array() 的参数

返回 二维数组（a list of lists）

flask_excel.ExcelRequest.**get_dict**(*field_name=None*, *sheet_name=None*, *name_columns_by_row=0*, ***keywords*)

参数

- **field_name** –和前面*get_sheet()* 一样。
- **sheet_name** –和前面*get_sheet()* 一样。
- **name_columns_by_row** –栏目名在哪一样。缺省的话，默认栏目在第一行。
- **keywords** –其他 pyexcel.get_dict() 的参数

返回 字典

flask_excel.ExcelRequest.**get_records**(*field_name=None*, *sheet_name=None*, *name_columns_by_row=0*, ***keywords*)

参数

- **field_name** –和前面*get_sheet()* 一样。
- **sheet_name** –和前面*get_sheet()* 一样。
- **name_columns_by_row** –栏目名在哪一样。缺省的话，默认栏目在第一行。
- **keywords** –其他 pyexcel.get_records() 的参数

返回 字典列表 (a list of records)

flask_excel.ExcelRequest.**get_book**(*field_name=None*, ***keywords*)

参数

- **field_name** –和前面*get_sheet()* 一样。
- **keywords** –其他 pyexcel.get_book() 的参数

返回 pyexcel.Book

flask_excel.ExcelRequest.**get_book_dict**(*field_name=None*, ***keywords*)

参数

- **field_name** –和前面*get_sheet()*一样。
- **keywords** –其他 pyexcel.get_book_dict() 的参数

返回 以二维数组为值的字典 (dict of a list of lists)

flask_excel.ExcelRequest.**save_to_database**(*field_name=None*, *session=None*, *table=None*, *initializer=None*, *mapdict=None ** keywords*)

参数

- **field_name** –和前面*get_sheet()*一样。
- **session** –SQLAlchemy 的 session
- **table** –数据库的一个表
- **initializer** –自定义的初始化函数
- **mapdict** –表栏目适配字典
- **keywords** –参照 pyexcel.Sheet.save_to_database()

flask_excel.ExcelRequest.**isave_to_database**(*field_name=None*, *session=None*, *table=None*, *initializer=None*, *mapdict=None ** keywords*)

和*save_to_database()*一样但用更少的内存。

同时要求上传文件的第一行是栏目名。

flask_excel.ExcelRequest.**save_book_to_database**(*field_name=None*, *session=None*, *tables=None*, *initializers=None*, *mapdicts=None*, ***keywords*)

参数

- **field_name** –和前面*get_sheet()*一样。
- **session** –SQLAlchemy 的 session
- **tables** –一组数据库表
- **initializers** –一组自定义的初始化函数
- **mapdicts** –一组表栏目适配字典。请注意，数据库表，初始化函数和栏目适配字典需要一一对应。
- **keywords** –参照 pyexcel.Book.save_to_database()

flask_excel.ExcelRequest.**isave_book_to_database**(*field_name=None*, *session=None*, *tables=None*, *initializers=None*, *mapdicts=None*, ***keywords*)

和*save_book_to_database()*. 一样但需要更少的内存

同时要求上传文件的所有表的第一行是栏目名。

10.2 Response methods

flask_excel.**make_response**(*pyexcel_instance*, *file_type*, *status=200*, *file_name=None*)

参数

- **pyexcel_instance** –pyexcel.Sheet 或 pyexcel.Book
- **file_type** –任何一个支持的文件类型，以下是可用的但不局限于它们的集合
  - ’csv’
  - ’tsv’
  - ’csvz’
  - ’tsvz’
  - ’xls’
  - ’xlsx’
  - ’xlsm’
  - ’ods’
- **status** –允许开发人员发自定义的 http status
- **file_name** –自定义的下载文件名称。注意，文件扩展名是不能改变的。

flask_excel.**make_response_from_array**(*array*, *file_type*, *status=200*, *file_name=None*)

参数

- **array** –二维数组（a list of lists）
- **file_type** –和*make_response()* 一样
- **status** –和*make_response()* 一样
- **file_name** –和*make_response()* 一样

flask_excel.**make_response_from_dict**(*dict*, *file_type*, *status=200*, *file_name=None*)

参数

- **dict** –字典 (dict)
- **file_type** –和*make_response()* 一样
- **status** –和*make_response()* 一样
- **file_name** –和*make_response()* 一样

flask_excel.**make_response_from_records**(*records*, *file_type*, *status=200*, *file_name=None*)

参数

- **records** –字典列表（records)
- **file_type** –和*make_response()* 一样
- **status** –和*make_response()* 一样
- **file_name** –和*make_response()* 一样

flask_excel.**make_response_from_book_dict**(*book_dict*, *file_type*, *status=200*, *file_name=None*)

参数

- **book_dict** –以二维数组为值的字典 (a dict of a list of lists)
- **file_type** –和*make_response()* 一样
- **status** –和*make_response()* 一样
- **file_name** –和*make_response()* 一样

flask_excel.**make_response_from_a_table**(*session*, *table*, *file_type*, *status=200*, *file_name=None*)

产生一个单页的 Excel 文件。里面的数据来自指定的数据库表。

参数

- **session** –SQLAlchemy 的 session
- **table** –数据库表
- **file_type** –same as *make_response()*
- **status** –same as *make_response()*
- **file_name** –same as *make_response()*

flask_excel.**make_response_from_query_sets**(*query_sets*, *column_names*, *file_type*, *status=200*, *file_name=None*)

产生一个单页的 Excel 文件。里面的数据来自查询结果。

参数

- **query_sets** –查询结果

- **column_names** –指定的栏目名字。如果是 None 的话，不会有数据返回哦。
- **file_type** –和*make_response()* 一样
- **status** –和*make_response()* 一样
- **file_name** –和*make_response()* 一样

flask_excel.**make_response_from_tables**(*session*, *tables*, *file_type*, *status=200*, *file_name=None*)

产生一个多页的 Excel 文件。如果 *tables* 里只有一个数据库表的话，它的功能就和*make_response_from_a_table()* 一样了。

参数

- **session** –SQLAlchemy 的 session
- **tables** –一组数据库表
- **file_type** –和*make_response()* 一样
- **status** –和*make_response()* 一样
- **file_name** –和*make_response()* 一样

Python 模块索引

f

索引

F

G

I

M

S

