django-excel Documentation

发布 *0.0.10*

Onni Software Ltd.

2021 年 07 月 10 日

Contents

作者 C.W.

源文件 http://github.com/pyexcel-webwares/django-excel.git

提交问题 http://github.com/pyexcel-webwares/django-excel/issues

发布的版本 0.0.10

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。"
```

django-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，**django-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 Django 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	hand-sontable	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	csv-totable	same as above
pyexcel-gantt	gantt chart in html	frappe-gantt	except pypy, same as above

既然有了 pyexcel , 为什么我还要装 django-excel ？

1. **加快上传文件处理速度.** **django-excel** 通过 `ExcelMemoryFileUploadHandler` 和 `TemporaryExcelFileUploadHandler` 让你直接处理上传的文件。*ExcelemoryFileUploadHandler* 把上传的文件存在内存里，这样呢 django-excel 可以直接从内存读取上传文件。*TemporaryExcelFileUpload-Handler* 呢，是把上传文件存在临时文件中，django-excel 则是直接读临时文件。到底是哪个类被 Django 调用呢？是由 FILE_UPLOAD_MAX_MEMORY_SIZE 决定的。如果上传文件小于这个最大值，那么 Django 会把上传文件存在内存里。否则，就存在临时文件里。

2. **直接把上传数据存入数据库**. **django-excel** 用批量方式（bulk_insert ）把你的数据导入你的 Django model.

CHAPTER 2

安装

你可以通过 pip 安装 django-excel：

```
$ pip install django-excel
```

或者复制到本地再安装：

```
$ git clone https://github.com/pyexcel-webwares/django-excel.git
$ cd django-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
```

一反 Django 开箱即用的理念，django-excel 需要开发人员自己选择所需 pyexcel 的套件。主要原因是，第三方软件 xlwt, openpyxl, odfpy 也是会拉长下载速度和占用磁盘空间。

CHAPTER 3

配置

You will need to update your *settings.py*:

```
FILE_UPLOAD_HANDLERS = ("django_excel.ExcelMemoryFileUploadHandler",
                        "django_excel.TemporaryExcelFileUploadHandler")
```

CHAPTER 4

测试过的 Django 版本

2.1, 2.0.8, 1.11.15, 1.10.8, 1.9.13, 1.8.18, 1.7.11, 1.6.11

从 2015 年 3 月 15 日之后，python 2.6 就再也没有测试过了

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

新手起步指南

为了让大家能轻松上手呢，作者给大家一步一步介绍 django-excel 自己的测试项目。所以，请先复制一下 '整个项目 <https://github.com/pyexcel/django-excel>'_ ，我们一起做:

```
git clone https://github.com/pyexcel/django-excel.git
```

整个测试项目是按照 Django 的指南的第 一, 二，三 部分写的。所以，作者就不最述了。如果大家想自己从零开始呢，请移步到 Django 指南，做完第三部分，再回来。

当你在本地有了项目代码呢，请进入 django-excel 目录并装上所有的软件包:

```
$ cd django-excel
$ pip install -r requirements.txt
$ pip install -r tests/requirements.txt
```

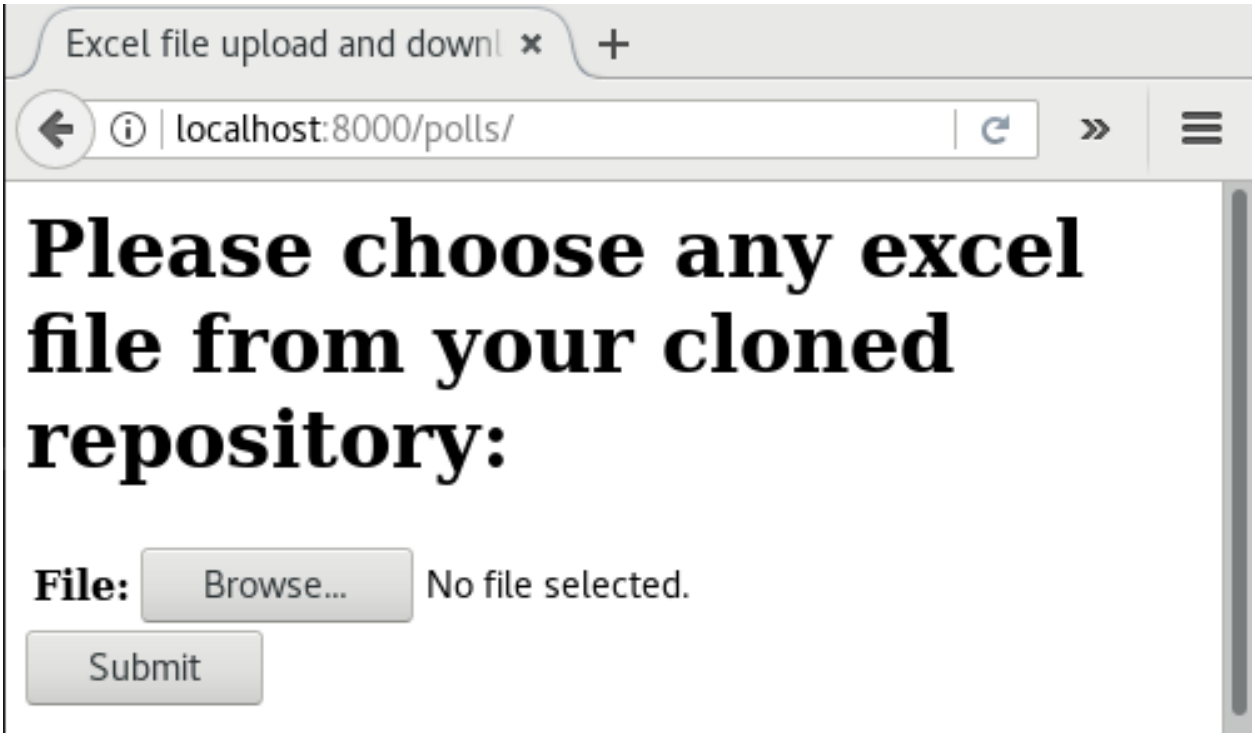
然后运行服务器:

```
$ python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 9 unapplied migration(s). Your project may not work properly until you apply␣
↪the migrations for app(s): admin, auth, contenttypes.
Run 'python manage.py migrate' to apply them.

July 06, 2017 - 08:29:10
Django version 1.11.3, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

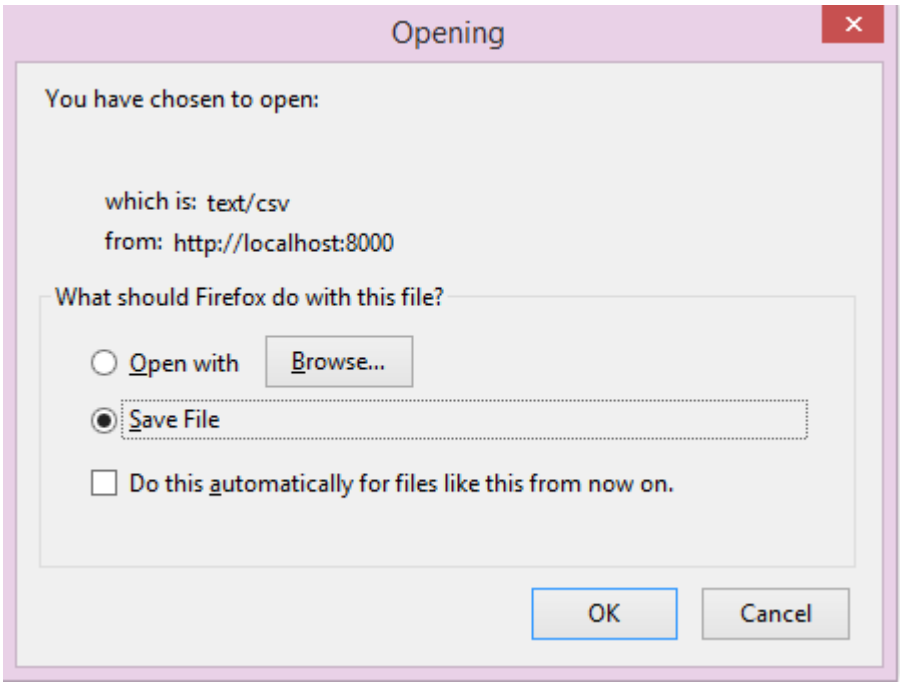
注解: 我们暂且忽略 9 未执行的迁徙。因为此指南的目的是让大家上手。

6.1 把上传的 excel 文件变成 csv 文件下载

我们来看看处理 excel 文件上传和 excel 文件下载的例子。用你的浏览器打开这个链接：http://localhost:8000/polls/, 你应该可以看到下面这个上传表格：



然后选择一个 xls 文件，然后按"Submit". 你会得到一个 csv 的下载文件。



我们看看源文件吧 polls/views.py :

```
# Create your views here.
def upload(request):
    if request.method == "POST":
        form = UploadFileForm(request.POST, request.FILES)
        if form.is_valid():
            filehandle = request.FILES["file"]
            return excel.make_response(
                filehandle.get_sheet(), "csv", file_name="download"
            )
    else:
        form = UploadFileForm()
    return render(
        request,
        "upload_form.html",
        {
            "form": form,
            "title": "Excel file upload and download example",
            "header": (
                "Please choose any excel file "
                + "from your cloned repository:"
            ),
        },
```

UploadFileForm 是 Django 的一个文件上传模块。然后往下看 **filehandle**. 它可能是 ExcelInMemoryUploadedFile 或是 TemporaryUploadedExcelFile。它们两个都继承了 ExcelMixin，所以它们都又有以下的函数，比如 get_sheet, get_array 。

把 excel 文件做为下载的函数, *make_response()* 能把由*get_sheet()* 得到 `pyexcel.Sheet` 实例转换成可以下载的 csv 文件。

你可以自己试试其他的函数*the mapping table*.

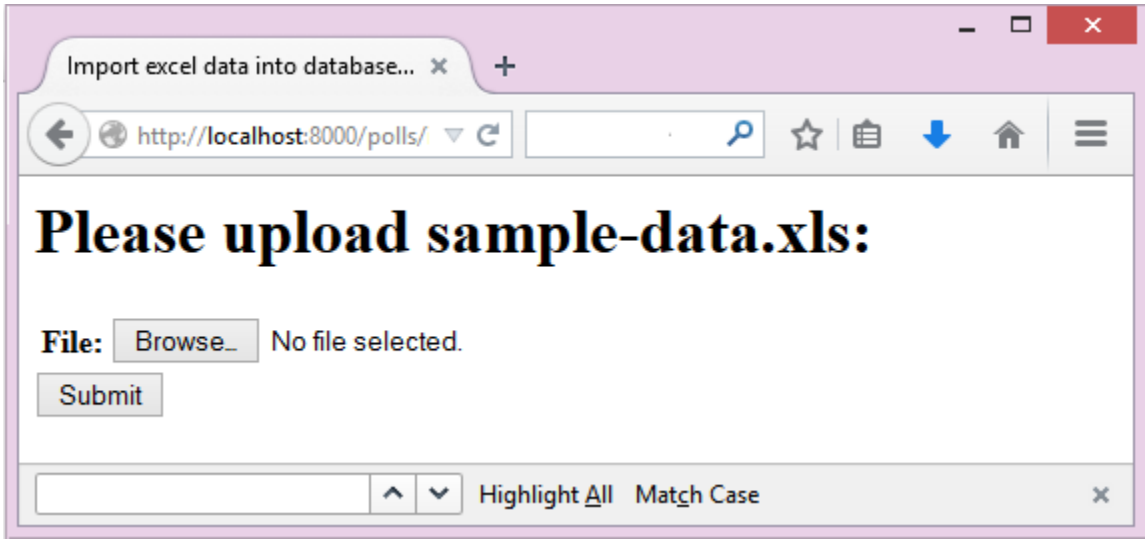
6.2 处理数据输入

这个例子展示如何把上传的 excel 的内容直接输入数据库。sample-data.xls 是我们要上传的文件。

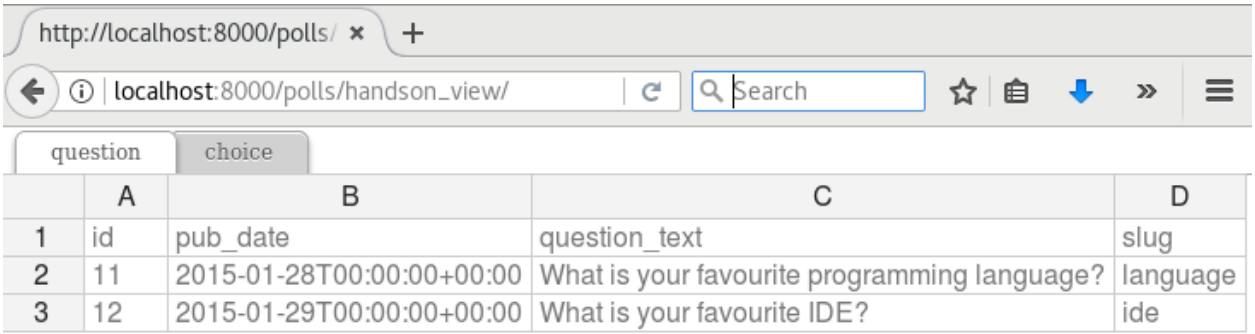
以下是我们的数据库 tables:

注解: 除了"slug" 属性, **Question** 和 **Choice** 都是从 Django 指南第一部分抄下来的。

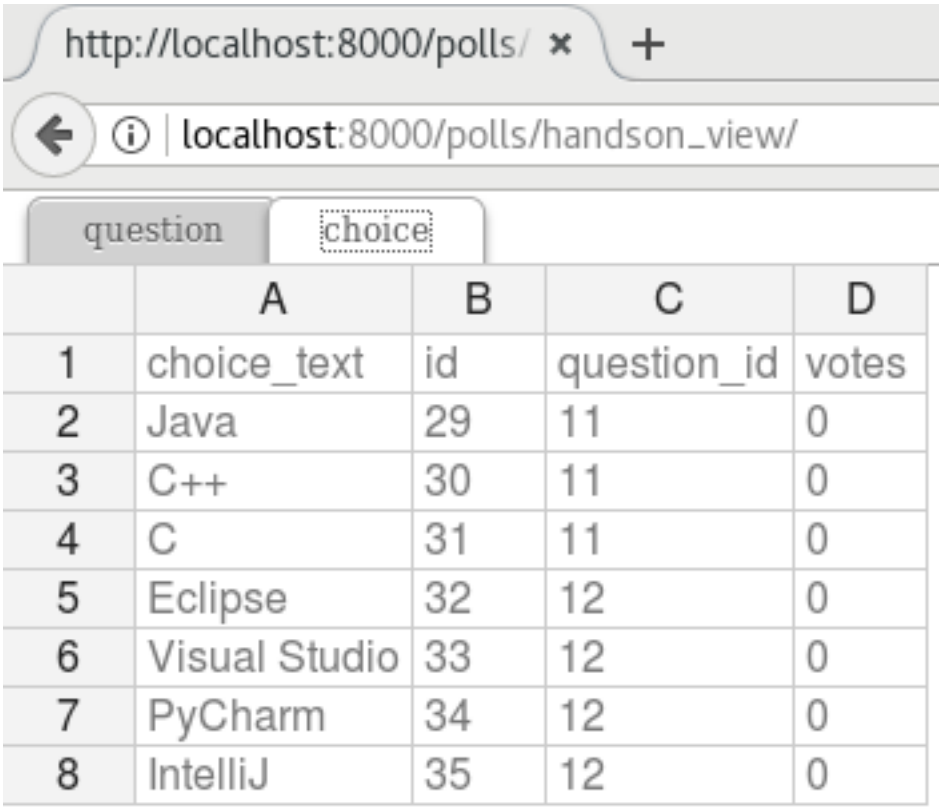
好了，打开链接 http://localhost:8000/polls/import/, 下面的文件上传页面：



再上传 sample-data.xls 。同时你会得到一个网页，里面显示了收到的数据。



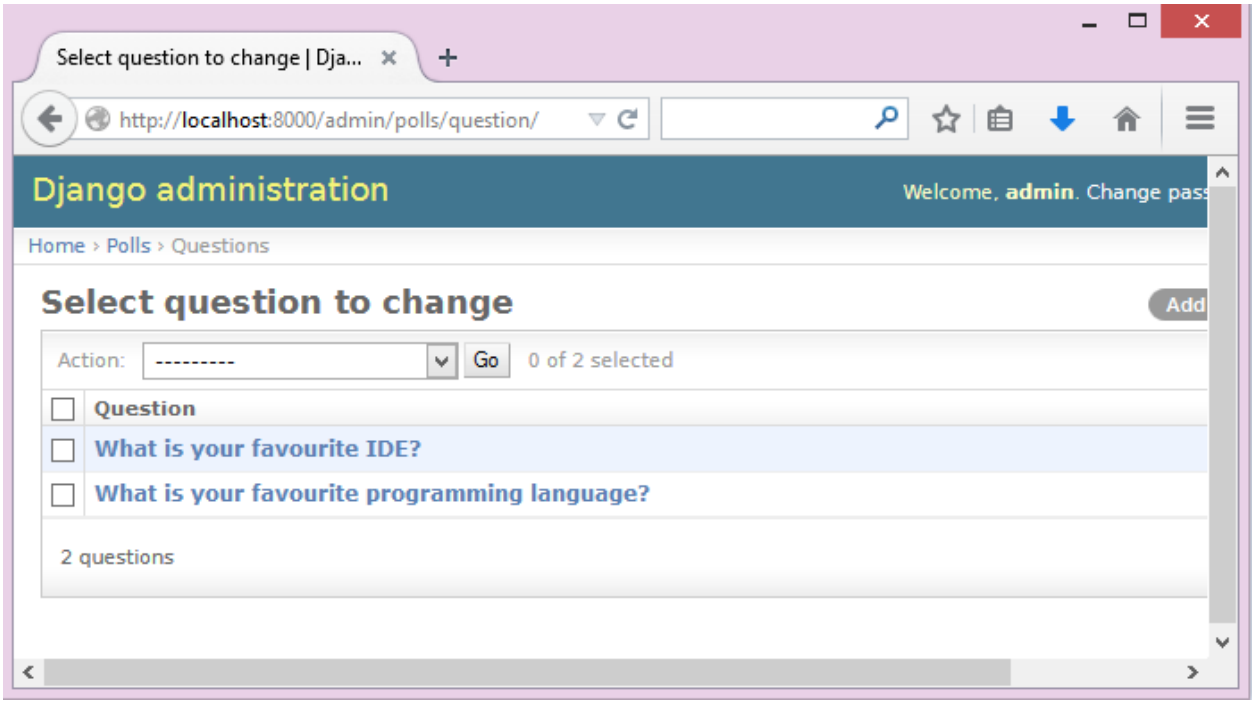
	A	B	C	D
1	id	pub_date	question_text	slug
2	11	2015-01-28T00:00:00+00:00	What is your favourite programming language?	language
3	12	2015-01-29T00:00:00+00:00	What is your favourite IDE?	ide



	A	B	C	D
1	choice_text	id	question_id	votes
2	Java	29	11	0
3	C++	30	11	0
4	C	31	11	0
5	Eclipse	32	12	0
6	Visual Studio	33	12	0
7	PyCharm	34	12	0
8	IntelliJ	35	12	0

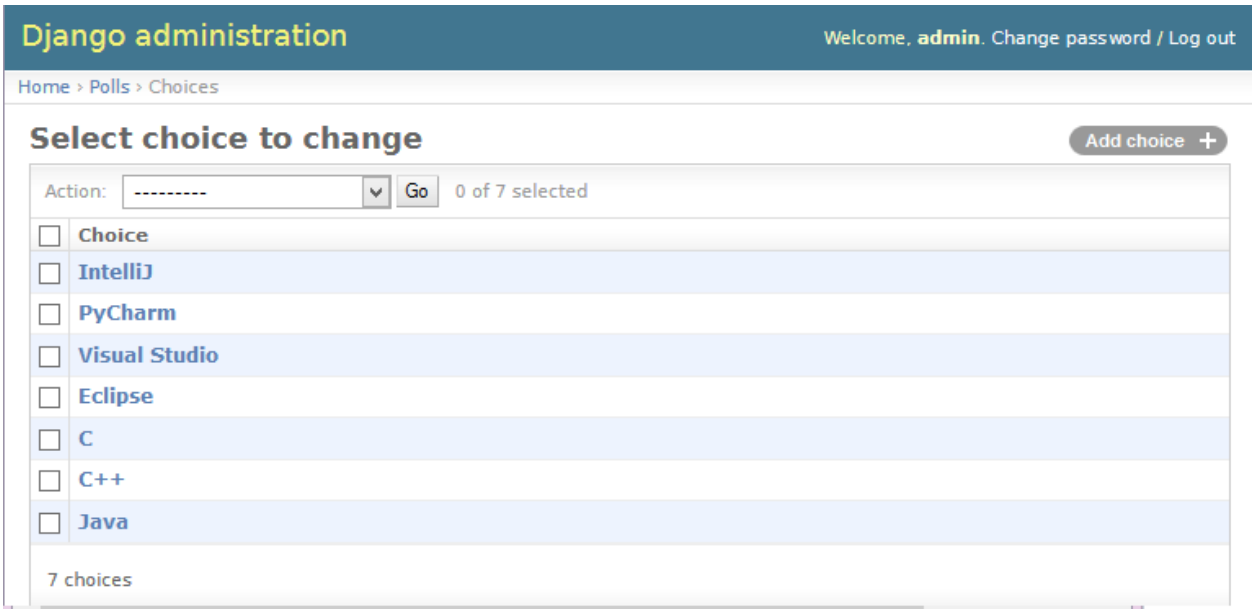
注解： pyexcel-handsontable 是 pyexcel 0.5.0 之后写的。它给用户带来 excel 一样的数据展示。在网站开发的时候，这个数据显示很上手。我后面会继续提到。

再去 Django 的管理员界面 http://localhost:8000/admin/polls/question, 你会发现 question 已经有数据了：



注解: 管理员界面的用户名是：admin。密码是: admin 。

再看看 choice:



注解: 你可以用管理员界面不数据都删了，再重复上传一次。

现在，我们来读一下源代码 polls/views.py 请关注这部分代码：

```
def import_data(request):
    if request.method == "POST":
        form = UploadFileForm(request.POST, request.FILES)

        def choice_func(row):
            q = Question.objects.filter(slug=row[0])[0]
            row[0] = q
            return row

        if form.is_valid():
            request.FILES["file"].save_book_to_database(
                models=[Question, Choice],
                initializers=[None, choice_func],
                mapdicts=[
                    ["question_text", "pub_date", "slug"],
                    {"Question": "question", "Choice": "choice_text", "Votes": "votes
↪"},
                ],
            )
            return redirect("handson_view")
        else:
            return HttpResponseBadRequest()
```

把上传的 excel 存入数据库的功臣是 `save_book_to_database()`. 函数变量 **models** 是 Django model 数组；**initializers** 是与之对应的初始化函数。在代码中，你会注意到，作者没有给 Question 写初始化函数，所以就给了 None；但是把 *choice_func* 给了 Choice。**mapdicts** 是一个用来控制数列栏名字的数组。它的成员可以是

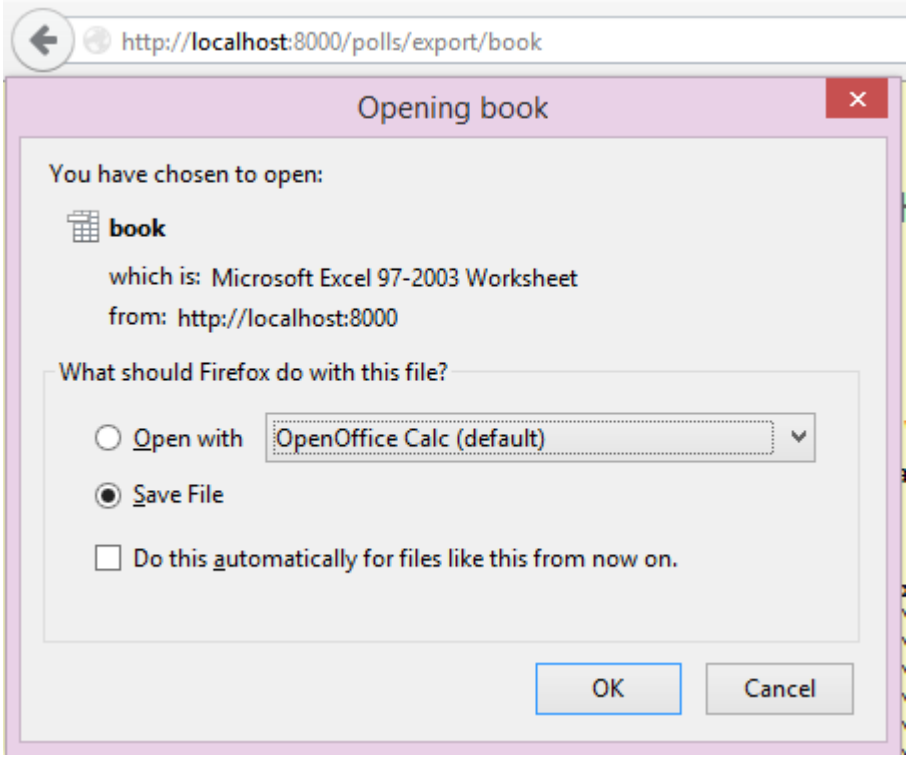
一个数组，也可以是一个字典:

```
{
  "Question Text": "question_text",
  "Publish Date": "pub_date",
  "Unique Identifier": "slug"
}
```

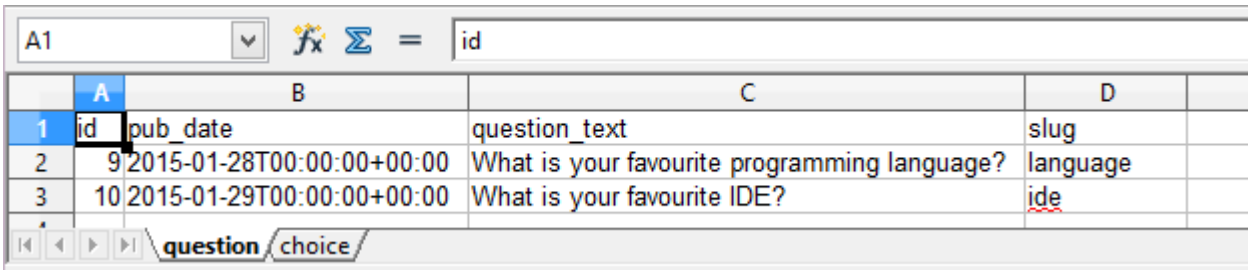
choice_func 是必须写的，因为 **Choice** 有一个外键对应 **Question**. **Choice** 的"Question" 属性需要已经存入数据库的一个主键。在我们的例子里，"Sheet 2" 的"Question" 一栏就必须和"Sheet 1" 的 Question 是一一对应的关系。

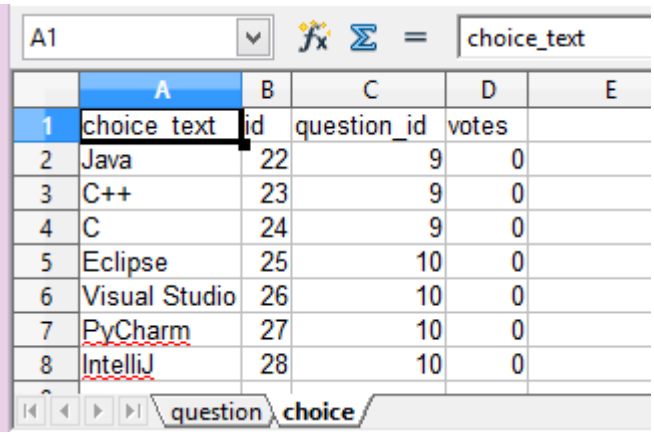
6.3 处理数据导出

我们再来具体看看如何把数据库里的表输出成 excel 文件。现在打开链接：http://localhost:8000/polls/export/book，这次呢，一个下载对话框直接就跳出来了:



把它存下来，然后打开看看。下载的数据看起来像这样:





	A	B	C	D	E
1	choice_text	id	question_id	votes	
2	Java	22	9	0	
3	C++	23	9	0	
4	C	24	9	0	
5	Eclipse	25	10	0	
6	Visual Studio	26	10	0	
7	PyCharm	27	10	0	
8	IntelliJ	28	10	0	

哪我们回头看看代码是怎么实现的: polls/views.py:

```
def export_data(request, atype):
    if atype == "sheet":
        return excel.make_response_from_a_table(
            Question, "xls", file_name="sheet"
        )
    elif atype == "book":
        return excel.make_response_from_tables(
            [Question, Choice], "xls", file_name="book"
```

make_response_from_tables() 其实做了所有的事情: 读取数据，把它们转换成 xls，再发给浏览器。做为开发人员，你需要给出要输出的表和 excel 文件类型。

与此同时，你还可打开另外一个链接：http://localhost:8000/polls/export/sheet 。它会把 **Question** 输出成一个单页的表格文件。

6.4 直接把 excel 文件渲染成 excel 的样子

最开始已经出现了这个渲染的形式。首先呢，handsontable 开发人员 做了所有的工作。其次 pyexcel-handsontable 仅仅做了集成而已。想要用这个的话，你需要自己装:

```
$ pip install pyexcel-handsontable
```

现在，我们看看这个是如何调用的。简单地说，就是把输出的文件后最写成: 'handsontable.html' 就可以了。

```
    return excel.make_response_from_tables(
        [Question, Choice], "handsontable.html"
    )
```

你可以把 handontable 文件嵌入 Django 的模版里面。下面是所需的代码:

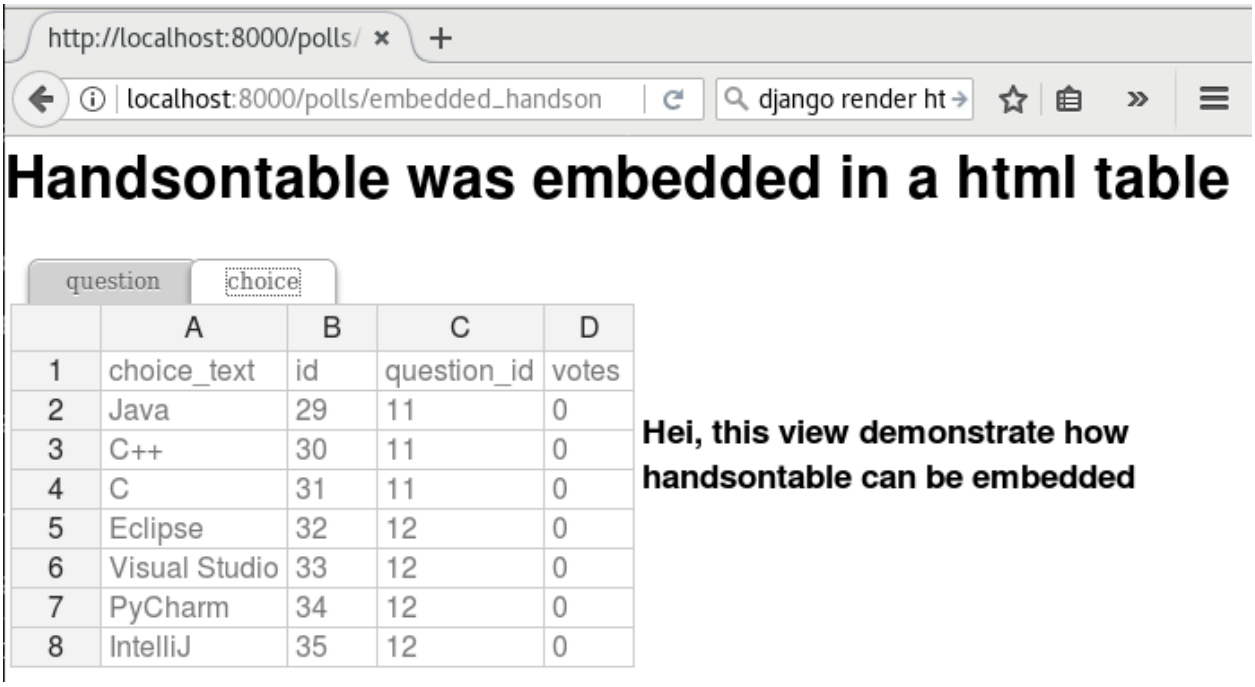
```
def embed_handson_table(request):
    """
    Renders two table in a handsontable
    """
    content = excel.pe.save_book_as(
        models=[Question, Choice],
        dest_file_type="handsontable.html",
        dest_embed=True,
```

(下页继续)

(续上页)

```
    )
    content.seek(0)
    return render(
        request,
        "custom-handson-table.html",
        {"handsontable_content": content.read()},
    )


def embed_handson_table_from_a_single_table(request):
    """
    Renders one table in a handsontable
    """
    content = excel.pe.save_as(
        model=Question, dest_file_type="handsontable.html", dest_embed=True
    )
    content.seek(0)
    return render(
        request,
        "custom-handson-table.html",
        {"handsontable_content": content.read()},
    )
```

你可以打开这两个链接预览一下：http://localhost:8000/polls/embedded_handson_view/ and http://localhost:8000/polls/embedded_handson_view_single/.



	A	B	C	D
1	choice_text	id	question_id	votes
2	Java	29	11	0
3	C++	30	11	0
4	C	31	11	0
5	Eclipse	32	12	0
6	Visual Studio	33	12	0
7	PyCharm	34	12	0
8	IntelliJ	35	12	0

6.4.1 如何输入单个表格呢

我们开始之前，请用 django 的管理员界面，清空 question 和 choice 。

前面讲了如何把一个多表格的 excel 文件里的数据输入数据库。现在我们看看输入一个表格。打开这个链接：http://localhost:8000/polls/imports_sheet/, w 这次上传 sample-sheet.xls 然后 Django 管理员界面可以查看是否有收到数据。

下面我们来读代码：

```
def import_sheet(request):
    if request.method == "POST":
        form = UploadFileForm(request.POST, request.FILES)
        if form.is_valid():
            request.FILES["file"].save_to_database(
                name_columns_by_row=2,
                model=Question,
                mapdict=["question_text", "pub_date", "slug"],
            )
            return HttpResponse("OK")
        else:
```

因为是单个表格，所以我们给一个 mapdict 参数并调用*save_to_database()* 来存到一个 Django 模型里。

看到了多了一个参数'name_columns_by_row' 吗？为什么需要它？是这样的，一般来讲，如果你的表格的第一行是栏目名字呢，你就不需要它。在这个示例里，栏目名故意放在了第二行。你可以打开 sample-sheet.xls 查看一下。

注解： 如果你忘了清空数据的话呢，你会得到一下的错误输出：

```
Warning: Bulk insertion got below exception. Trying to do it one by one slowly.
column slug is not unique <- reason
One row is ignored <- action
column slug is not unique
What is your favourite programming language?
One row is ignored
column slug is not unique
What is your favourite IDE?
```

这是因为数据库里已经有相同数据了，Django 就会报 IntegrityError. 具体请读 这部分 pyexcel-io 的代码, 和 django-excel 问题 2

为了除去这个警告呢，你需要在 Django 的管理员界面清空所有数据。然后再试一下。

6.4.2 如果 excel 数据里有一些与数据库数据重叠了，怎么办？

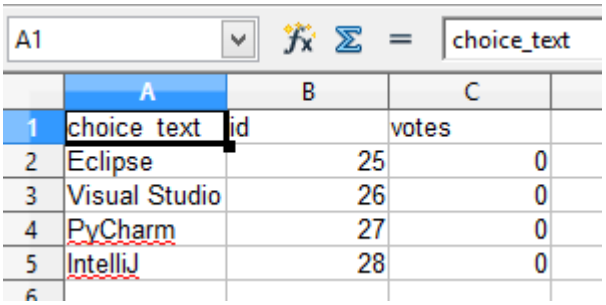
你可以提供自己的一个初始化函数。这个数在遇到重叠的数据的时候，返回 None 这样 django-excel 就会跳过当前的一行数据。当然，在初始化函数里，你也可以更新数据库。最重要的是，只有你的初始化函数返回 None，django-excel 会尽量用批量输入，而不是一个一个输入数据库。

6.5 处理自定义数据输出

有时候，应用户要求呢，我们会下载数据库表的一部分。这个时候，作为开发人员，你可以用*make_response_from_query_sets()* 来产生一个 excel 文件:

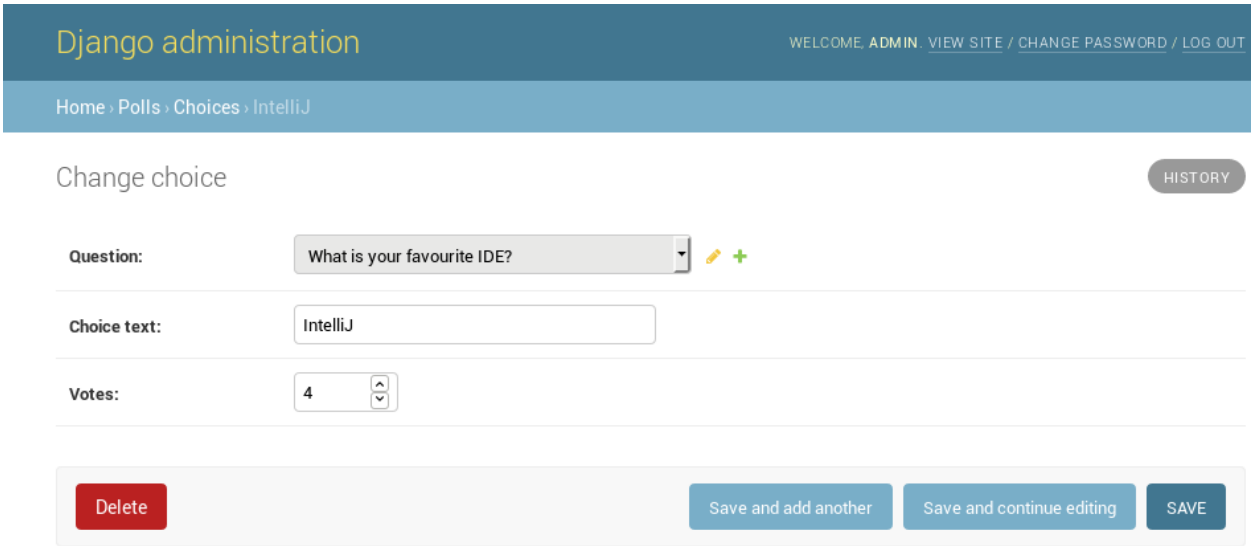
```
def export_data(request, atype):
            [Question, Choice], "xls", file_name="book"
        )
    elif atype == "custom":
        question = Question.objects.get(slug="ide")
        query_sets = Choice.objects.filter(question=question)
        column_names = ["choice_text", "id", "votes"]
        return excel.make_response_from_query_sets(
            query_sets, column_names, "xls", file_name="custom"
        )
    else:
```

你可以打开 http://localhost:8000/polls/export/custom 试试看看:

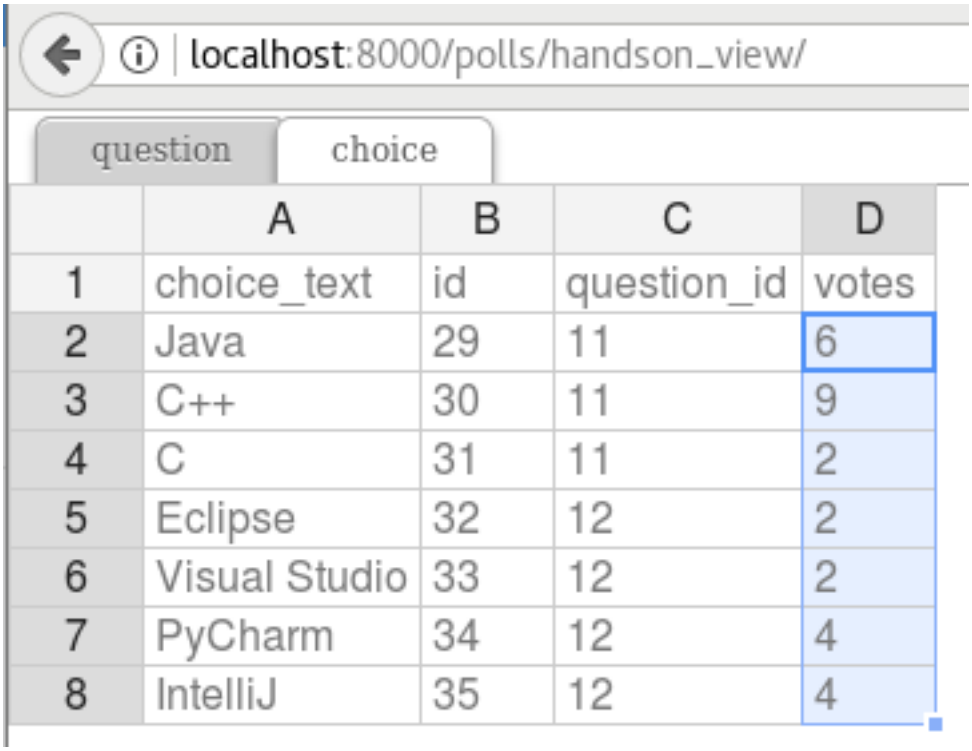


6.6 渲染你的数据

为了渲染数据，我们先去 Django 的管理员界面，增加一些投票。

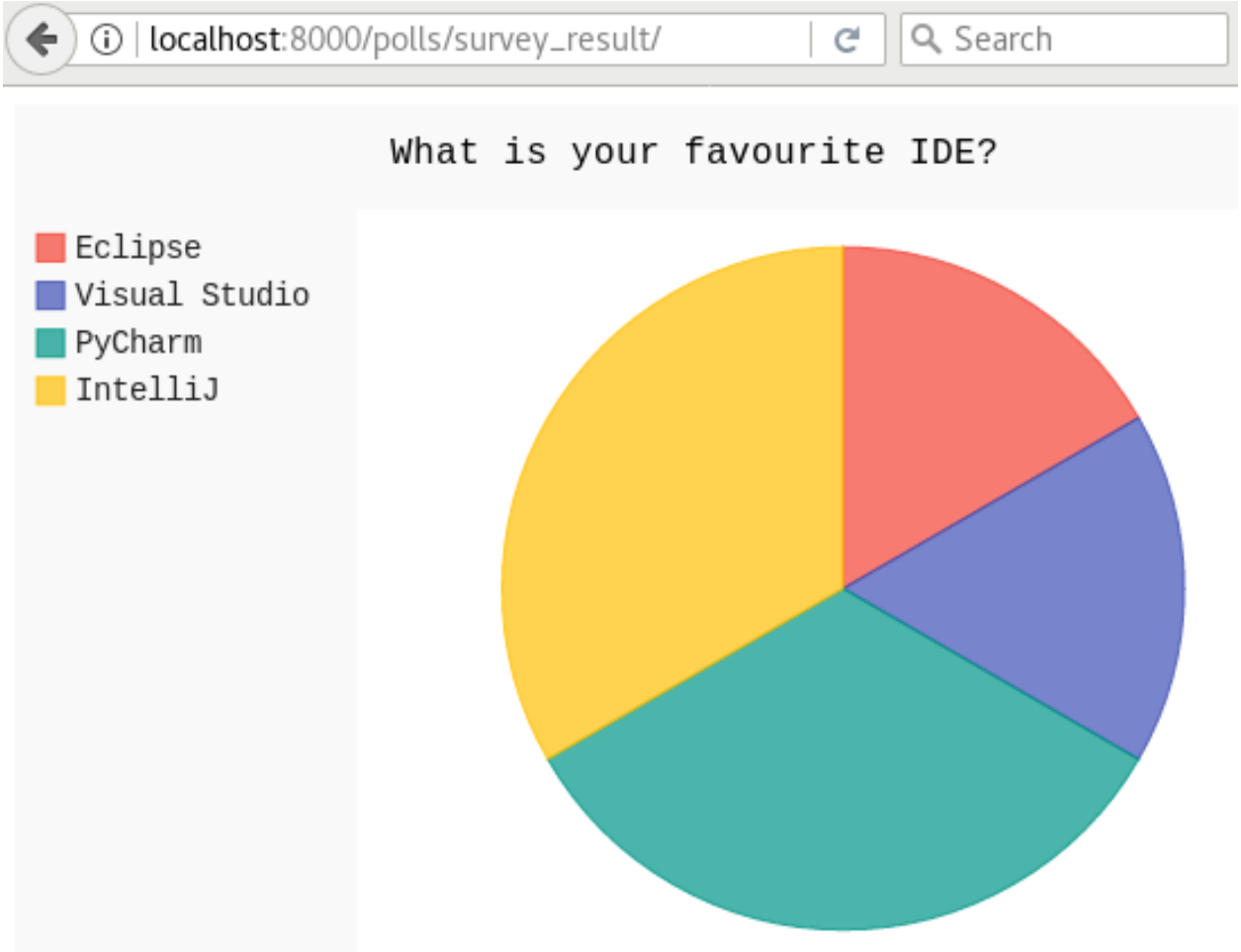


看看，这个是我改的:



	A	B	C	D
1	choice_text	id	question_id	votes
2	Java	29	11	6
3	C++	30	11	9
4	C	31	11	2
5	Eclipse	32	12	2
6	Visual Studio	33	12	2
7	PyCharm	34	12	4
8	IntelliJ	35	12	4

现在，我们来看看这个问卷调查的结果 (http://localhost:8000/polls/survey_result/) ”What's your favorite IDE?”:



pyexcel-pygal 是 pyexcel 的一个插件，可以用来展示一些常用图表。下面是所需要的代码：

```
    query_sets = Choice.objects.filter(question=question)
    column_names = ["choice_text", "votes"]

    # Obtain a pyexcel sheet from the query sets
    sheet = excel.pe.get_sheet(
        query_sets=query_sets, column_names=column_names
    )
    sheet.name_columns_by_row(0)
    sheet.column.format("votes", int)

    # Transform the sheet into an svg chart
    svg = excel.pe.save_as(
        array=[sheet.column["choice_text"], sheet.column["votes"]],
        dest_file_type="svg",
        dest_chart_type="pie",
        dest_title=question.question_text,
        dest_width=600,
        dest_height=400,
    )

    return render(request, "survey_result.html", dict(svg=svg.read()))
```

(下页继续)

(续上页)

```
def import_sheet_using_isave_to_database(request):
    if request.method == "POST":
        form = UploadFileForm(request.POST, request.FILES)
```

CHAPTER 7

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 8

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！

微信支付

CHAPTER 9

函数参考

django-excel 把 **pyexcel** 的函数嫁接到了 **InMemoryUploadedFile** 和 **TemporaryUploadedFile**。所以，以下的函数都会出现在上载文件的实例里：request.FILES['your_uploaded_file'] 。

`django_excel.ExcelMixin.`**`get_sheet`**(*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`

`django_excel.ExcelMixin.`**`get_array`**(*sheet_name=None*, ***keywords*)

参数

- **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）

`django_excel.ExcelMixin.`**`iget_array`**(*sheet_name=None*, ***keywords*)

参数

- **sheet_name** -- 和前面*`get_sheet()`* 一样。
- **keywords** -- 其他 `pyexcel.iget_array()` 的参数

返回 数组产生器

`django_excel.ExcelMixin.`**`get_dict`**(*sheet_name=None*, *name_columns_by_row=0*, ***keywords*)

参数

- **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

`django_excel.ExcelMixin.`**`get_records`**(*sheet_name=None*, *name_columns_by_row=0*, ***keywords*)

参数

- **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)

`django_excel.ExcelMixin.`**`iget_records`**(*sheet_name=None*, *name_columns_by_row=0*, ***keywords*)

参数

- **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.iget_records() 的参数

返回 字典产生器（a generator for records）

`django_excel.ExcelMixin.`**`get_book`**(***keywords*)

参数 **keywords** -- 其他 pyexcel.get_book() 的参数

返回 pyexcel.Book

`django_excel.ExcelMixin.`**`get_book_dict`**(***keywords*)

参数 **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)

`django_excel.ExcelMixin.`**`save_to_database`**(*model=None*, *initializer=None*, *mapdict=None*, ***keywords*)

参数

- **model** -- Django 模型
- **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_django_model()

`django_excel.ExcelMixin.`**`isave_to_database`**(*model=None*, *initializer=None*, *mapdict=None*, ***keywords*)

和*save_to_database()* 一样但需要更少内存

同时要求上传文件的第一行是栏目名。

`django_excel.ExcelMixin.`**`save_book_to_database`**(*models=None*, *initializers=None*, *mapdicts=None*, ***keywords*)

参数

- **models** -- Django 模型数组
- **initializers** -- 自定义的初始化函数组
- **mapdicts** -- 表栏目适配字典组
- **keywords** -- 参照 `pyexcel.Book.save_to_django_models()`

此函数与 `save_to_database()` 类似，只不过是单数边多数而已。

`django_excel.ExcelMixin.`**`isave_book_to_database`**(*models=None*, *initializers=None*, *mapdicts=None*, ***keywords*)

和*`save_book_to_database()`* 一样，但需要更少内存。

同时要求上传文件的所有表的第一行是栏目名。

`django_excel.ExcelMixin.`**`free_resources`**()

如果用了 *iget_array* 和 *iget_reords*, 这个需要调用

CHAPTER 10

回复函数

`django_excel.`**`make_response`**(*pyexcel_instance*, *file_type*, *status=200*)

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

`django_excel.`**`make_response_from_array`**(*array*, *file_type*, *status=200*)

参数

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

`django_excel.`**`make_response_from_dict`**(*dict*, *file_type*, *status=200*)

参数 **dict** -- a dictionary of lists

:param file_type: 和`make_response()` 一样:param status: 和`make_response()` 一样

`django_excel.`**`make_response_from_records`**(*records*, *file_type*, *status=200*)

参数

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

`django_excel.`**`make_response_from_book_dict`**(*book_dict*, *file_type*, *status=200*)

参数

- **book_dict** -- 以二维数组为值的字典 (a dictionary of two dimensional arrays）
- **file_type** -- 和`make_response()` 一样
- **status** -- 和`make_response()` 一样

`django_excel.`**`make_response_from_a_table`**(*model*, *file_type status=200*)
Produce a single sheet Excel book of *file_type*

参数

- **model** -- 一个 Django 模型
- **file_type** -- 和`make_response()` 一样
- **status** -- 和`make_response()` 一样

`django_excel.`**`make_response_from_query_sets`**(*query_sets*, *column_names*, *file_type status=200*)
由询问结果产生 *file_type* 类的 Excel 文件

参数

- **query_sets** -- 询问结果
- **column_names** -- a nominated column names. It could not be None, otherwise no data is returned.
- **file_type** -- 和`make_response()` 一样
- **status** -- 和`make_response()` 一样

`django_excel.`**`make_response_from_tables`**(*models*, *file_type status=200*)
产生一个多页的 Excel 文件。和`make_response_from_a_table()` 类似。

参数

- **models** -- Django 模型组
- **file_type** -- 和`make_response()` 一样
- **status** -- 和`make_response()` 一样

Python 模块索引

d

索引