django-tabular-export Documentation

Release 1.0.0

Chris Adams

March 04, 2016

Contents

Contents:

CHAPTER 1

django-tabular-export

Simple spreadsheet exports from Django

1.1 Documentation

This module contains functions which take (headers, rows) pairs and return HttpResponses with either XLSX or CSV downloads and Django admin actions which can be added to any ModelAdmin for generic exports. It provides two functions (`export_to_csv_response` and `export_to_xlsx_response`) which take a filename, a list of column headers, and a Django `QuerySet`, list-like object, or generator and return a response.

1.1.1 Goals

- This project is not intended to be a general-purpose spreadsheet manipulation library. The only goal is to export data quickly and safely.
- The API is intentionally simple, giving you full control over the display and formatting of headers or your data. `flatten_queryset` has special handling for only two types of data: `None` will be converted to an empty string and `date` or `datetime` instances will serialized using `isoformat()`. All other values will be specified as the text data type to avoid data corruption in Excel if the values happen to resemble a date in the current locale.
- **Unicode-safety**: input values, including lazy objects, are converted using Django's force_text function and will always be emitted as UTF-8
- **Performance**: the code is known to work with data sets up to hundreds of thousands of rows. CSV responses use `StreamingHttpResponse`, use minimal memory, and start very quickly. Excel (XLSX) responses cannot be streamed but xlsxwriter is one of the faster implementations and its memory-size optimizations are enabled.

1.2 Quickstart

Install django-tabular-export:

```
pip install django-tabular-export
```

Then use it in a project:

```
from tabular_export import export_to_csv_response, export_to_xlsx_response, flatten_queryset

def my_view(request):
    return export_to_csv_response('test.csv', ['Column 1'], [['Data 1'], ['Data 2'], ...])


def my_other_view(request):
    headers = ['Title', 'Date Created']
    rows = MyModel.objects.values_list('title', 'date_created')
    return export_to_excel_response('items.xlsx', headers, rows)


def export_using_a_generator(request):
    headers = ['A Number']

    def my_generator():
        for i in range(0, 100000):
            yield (i, )

    return export_to_excel_response('numbers.xlsx', headers, my_generator())

def export_renaming_columns(request)
    qs = MyModel.objects.filter(...).select_related(...)
    headers, data = flatten_queryset(qs, field_names=['title', 'related_model__title_en'],
                                     extra_verbose_names={'related_model__title_en': 'English Title'}
    return export_to_csv_response('custom_export.csv', headers, data)
```

1.2.1 Admin Integration

There are two convenience admin actions which make it simple to add "Export to Excel" and "Export to CSV" actions:

```
from tabular_export.admin import export_to_csv_action, export_to_excel_action

class MyModelAdmin(admin.ModelAdmin):
    actions = (export_to_excel_action, export_to_csv_action)
    ...
```

The default columns will be the same as you would get calling `values_list` on your `ModelAdmin`'s default queryset as returned by `ModelAdmin.get_queryset()`. If you want to customize this, simply declare a new action on your `ModelAdmin` which does whatever data preparation is necessary:

```
from tabular_export.admin import export_to_excel_action

class MyModelAdmin(admin.ModelAdmin):
    actions = ('export_batch_summary_action', )

    def export_batch_summary_action(self, request, queryset):
        headers = ['Batch Name', 'My Computed Field']
        rows = queryset.annotate(...).values_list('title', 'computed_field_name')
        return export_to_excel_response('batch-summary.xlsx', headers, rows)
    export_batch_summary_action.short_description = 'Export Batch Summary'
```

1.2.2 Debugging

The `TABULAR_RESPONSE_DEBUG = True` setting will cause all views to return HTML tables

CHAPTER 2

tabular_export

2.1 tabular_export package

2.1.1 Submodules

tabular_export.admin module

Usage can be as simple as adding the generic actions to a ModelAdmin:

```
actions = (export_to_excel_action, export_to_csv_action)
```

These will take the QuerySet and provide a generic export action which is essentially what you'd from the `values()` method. The filename will be generated from the model name specified for that *ModelAdmin*.

The allow you to pass a custom file filename or list of fields which are passed through directly to `flatten_queryset()` and `export_to_excel_response()` / `export_to_csv_response()`

`tabular_export.admin.`**`ensure_filename`**(*suffix*)
Decorator which automatically sets the filename going into the admin actions from the `ModelAdmin.model`'s `verbose_name_plural` value unless a value was provided by the caller.

`tabular_export.admin.`**`export_to_csv_action`**(*modeladmin*, *request*, *queryset*, *filename=None*, **args*, ***kwargs*)
Django admin action which exports the selected records as a CSV download

`tabular_export.admin.`**`export_to_excel_action`**(*modeladmin*, *request*, *queryset*, *filename=None*, **args*, ***kwargs*)
Django admin action which exports selected records as an Excel XLSX download

tabular_export.core module

Exports to tabular (2D) formats

This module contains functions which take (headers, rows) pairs and return HttpResponses with either XLSX or CSV downloads

The `export_to_FORMAT_response` functions accept a `filename`, and `headers` and `rows`. This allows full control over the data using non-database data-sources, the Django ORM's various aggregations and optimization methods, generators for large responses, control over the column names, or post-processing using methods like `get_FOO_display()` to format the data for display.

The `flatten_queryset` utility used to generate lists from QuerySets intentionally does not attempt to handle foreign-key fields to avoid performance issues. If you need to include such data, prepare it in advance using whatever optimizations are possible and pass the data in directly.

If your Django settings module sets `TABULAR_RESPONSE_DEBUG` to `True` the data will be dumped as an HTML table and will not be delivered as a download.

class `tabular_export.core.`**Echo**
Bases: `object`

write(*value*)

`tabular_export.core.`**convert_value_to_unicode**(*v*)
Return the UTF-8 bytestring representation of the provided value

date/datetime instances will be converted to ISO 8601 format None will be returned as an empty string

`tabular_export.core.`**export_to_csv_response**(*filename*, **args*, ***kwargs*)
Returns a downloadable StreamingHttpResponse using an CSV payload generated from headers and rows

`tabular_export.core.`**export_to_debug_html_response**(*filename*, *headers*, *rows*)
Returns a downloadable StreamingHttpResponse using an HTML payload for debugging

`tabular_export.core.`**export_to_excel_response**(*filename*, **args*, ***kwargs*)
Returns a downloadable HttpResponse using an XLSX payload generated from headers and rows

`tabular_export.core.`**flatten_queryset**(*qs*, *field_names=None*, *extra_verbose_names=None*)
Return a tuple of named column headers and a list of data values

By default headers will use the keys from `qs.values()` and rows will use the more-efficient `values_list()`.

If a list of `field_names` are passed, only the included fields will be returned.

An optional dictionary of `extra_verbose_names` may be passed to provide friendly names for fields and will override the field's `verbose_name` attribute if present. This can be used to provide proper names for related lookups (e.g. *{"institution__title": "Institution"}*) or calculated values (*e.g. {"items__count": "Item Count"}*).

`tabular_export.core.`**force_utf8_encoding**(*f*)

`tabular_export.core.`**get_field_names_from_queryset**(*qs*)
Return a list of field names for a queryset, including extra and aggregate columns

`tabular_export.core.`**return_debug_reponse**(*f*)
Returns a debugging-friendly HTML response when TABULAR_RESPONSE_DEBUG is set

`tabular_export.core.`**set_content_disposition**(*f*)
Ensure that an HttpResponse has the Content-Disposition header set using the input filename= kwarg

2.1.2 Module contents

CHAPTER 3

License

```
Creative Commons Legal Code

CC0 1.0 Universal

    CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE
    LEGAL SERVICES. DISTRIBUTION OF THIS DOCUMENT DOES NOT CREATE AN
    ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS
    INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO WARRANTIES
    REGARDING THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS
    PROVIDED HEREUNDER, AND DISCLAIMS LIABILITY FOR DAMAGES RESULTING FROM
    THE USE OF THIS DOCUMENT OR THE INFORMATION OR WORKS PROVIDED
    HEREUNDER.

Statement of Purpose

The laws of most jurisdictions throughout the world automatically confer
exclusive Copyright and Related Rights (defined below) upon the creator
and subsequent owner(s) (each and all, an "owner") of an original work of
authorship and/or a database (each, a "Work").

Certain owners wish to permanently relinquish those rights to a Work for
the purpose of contributing to a commons of creative, cultural and
scientific works ("Commons") that the public can reliably and without fear
of later claims of infringement build upon, modify, incorporate in other
works, reuse and redistribute as freely as possible in any form whatsoever
and for any purposes, including without limitation commercial purposes.
These owners may contribute to the Commons to promote the ideal of a free
culture and the further production of creative, cultural and scientific
works, or to gain reputation or greater distribution for their Work in
part through the use and efforts of others.

For these and/or other purposes and motivations, and without any
expectation of additional consideration or compensation, the person
associating CC0 with a Work (the "Affirmer"), to the extent that he or she
is an owner of Copyright and Related Rights in the Work, voluntarily
elects to apply CC0 to the Work and publicly distribute the Work under its
terms, with knowledge of his or her Copyright and Related Rights in the
Work and the meaning and intended legal effect of CC0 on those rights.

1. Copyright and Related Rights. A Work made available under CC0 may be
protected by copyright and related or neighboring rights ("Copyright and
Related Rights"). Copyright and Related Rights include, but are not
limited to, the following:
```

```
  i. the right to reproduce, adapt, distribute, perform, display,
     communicate, and translate a Work;
 ii. moral rights retained by the original author(s) and/or performer(s);
iii. publicity and privacy rights pertaining to a person's image or
     likeness depicted in a Work;
 iv. rights protecting against unfair competition in regards to a Work,
     subject to the limitations in paragraph 4(a), below;
  v. rights protecting the extraction, dissemination, use and reuse of data
     in a Work;
 vi. database rights (such as those arising under Directive 96/9/EC of the
     European Parliament and of the Council of 11 March 1996 on the legal
     protection of databases, and under any national implementation
     thereof, including any amended or successor version of such
     directive); and
vii. other similar, equivalent or corresponding rights throughout the
     world based on applicable law or treaty, and any national
     implementations thereof.

2. Waiver. To the greatest extent permitted by, but not in contravention
of, applicable law, Affirmer hereby overtly, fully, permanently,
irrevocably and unconditionally waives, abandons, and surrenders all of
Affirmer's Copyright and Related Rights and associated claims and causes
of action, whether now known or unknown (including existing as well as
future claims and causes of action), in the Work (i) in all territories
worldwide, (ii) for the maximum duration provided by applicable law or
treaty (including future time extensions), (iii) in any current or future
medium and for any number of copies, and (iv) for any purpose whatsoever,
including without limitation commercial, advertising or promotional
purposes (the "Waiver"). Affirmer makes the Waiver for the benefit of each
member of the public at large and to the detriment of Affirmer's heirs and
successors, fully intending that such Waiver shall not be subject to
revocation, rescission, cancellation, termination, or any other legal or
equitable action to disrupt the quiet enjoyment of the Work by the public
as contemplated by Affirmer's express Statement of Purpose.

3. Public License Fallback. Should any part of the Waiver for any reason
be judged legally invalid or ineffective under applicable law, then the
Waiver shall be preserved to the maximum extent permitted taking into
account Affirmer's express Statement of Purpose. In addition, to the
extent the Waiver is so judged Affirmer hereby grants to each affected
person a royalty-free, non transferable, non sublicensable, non exclusive,
irrevocable and unconditional license to exercise Affirmer's Copyright and
Related Rights in the Work (i) in all territories worldwide, (ii) for the
maximum duration provided by applicable law or treaty (including future
time extensions), (iii) in any current or future medium and for any number
of copies, and (iv) for any purpose whatsoever, including without
limitation commercial, advertising or promotional purposes (the
"License"). The License shall be deemed effective as of the date CC0 was
applied by Affirmer to the Work. Should any part of the License for any
reason be judged legally invalid or ineffective under applicable law, such
partial invalidity or ineffectiveness shall not invalidate the remainder
of the License, and in such case Affirmer hereby affirms that he or she
will not (i) exercise any of his or her remaining Copyright and Related
Rights in the Work or (ii) assert any associated claims and causes of
action with respect to the Work, in either case contrary to Affirmer's
express Statement of Purpose.

4. Limitations and Disclaimers.
```

```
a. No trademark or patent rights held by Affirmer are waived, abandoned,
   surrendered, licensed or otherwise affected by this document.
b. Affirmer offers the Work as-is and makes no representations or
   warranties of any kind concerning the Work, express, implied,
   statutory or otherwise, including without limitation warranties of
   title, merchantability, fitness for a particular purpose, non
   infringement, or the absence of latent or other defects, accuracy, or
   the present or absence of errors, whether or not discoverable, all to
   the greatest extent permissible under applicable law.
c. Affirmer disclaims responsibility for clearing rights of other persons
   that may apply to the Work or any use thereof, including without
   limitation any person's Copyright and Related Rights in the Work.
   Further, Affirmer disclaims responsibility for obtaining any necessary
   consents, permissions or other rights required for any use of the
   Work.
d. Affirmer understands and acknowledges that Creative Commons is not a
   party to this document and has no duty or obligation with respect to
   this CC0 or use of the Work.
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/LibraryOfCongress/django-tabular-export/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

4.1.4 Write Documentation

django-tabular-export could always use more documentation, whether as part of the official django-tabular-export docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/LibraryOfCongress/django-tabular-export/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *django-tabular-export* for local development.

1. Fork the *django-tabular-export* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-tabular-export.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-tabular-export
$ cd django-tabular-export/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 tabular_export tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/LibrLibraryOfCongressCongress/django-tabular-export/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_tabular_export
```

CHAPTER 5

Credits

5.1 Development Lead

- Chris Adams <cadams@loc.gov>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

Changelog

6.1 v1.0.1 (2016-03-04)

6.1.1 Fix

- CSV UTF-8 regression introduced with Python 3 support. [Chris Adams]

  The refactor to add Python 3 support to the long-running Python 2 version introduced a regression for Unicode handling with CSV output.

6.2 v1.0.0 (2016-03-04)

- Initial Release. [Chris Adams]

t

C

E

F

G

R

S

T

W