

---

# **django-model-report Documentation**

*Release 0.2.1*

**juanpex**

**Jun 28, 2017**



---

## Contents

---

|                            |          |
|----------------------------|----------|
| <b>1 Demo</b>              | <b>3</b> |
| 1.1 User Guide . . . . .   | 3        |
| 1.2 Modules . . . . .      | 4        |
| <b>Python Module Index</b> | <b>9</b> |



django-model-report is a Django application and library for reports integrated with highcharts.



<http://django-model-report.herokuapp.com>

## User Guide

### Installation

django-model-report is on the Python Package Index (PyPI), so it can be installed with standard Python tools like pip or easy\_install.:

```
$ pip install django-model-report
```

- Add the `model_report` directory to your Python path.
- Add `model_report` to your `INSTALLED_APPS` setting.
- Create file “reports.py” within any application directory (just like `admin.py`).
- Edit the file “reports.py” and register your reports like this:

```
from anyapp.models import AnyModel
from model_report.report import reports, ReportAdmin

class AnyModelReport (ReportAdmin):
    title = _('AnyModel Report Name')
    model = AnyModel
    fields = [
        'anymodelfield',
    ]
    list_order_by = ('anymodelfield',)
    type = 'report'

reports.register('anymodel-report', AnyModelReport)
```

- Activate your reports calling the autodiscover in `urls.py` (just like `admin.py`):

```
from model_report import report
report.autodiscover()
```

- Add reports urls.:

```
urlpatterns = patterns('',
    ...
    (r'', include('model_report.urls')),
    ...
)
```

## Example app

```
cd example && ./manage.py runserver
```

Username and password for admin are 'admin', 'admin'.

## Configuration

Extend your reports from `model_report.report.ReportAdmin()`:

```
from model_report.report import ReportAdmin

class ReportExample(ReportAdmin):
    pass
```

To configure the report see the `model_report.report.ReportAdmin()` documentation

## Contributing

Clone the repo and help to be better this app :)

## Modules

### Modules

#### highcharts

##### Highchart Module

###### `__init__`

`model_report.highcharts.__init__.HTMLEntitiesToUnicode` (*text*)  
Converts HTML entities to unicode. For example '&amp;' becomes '&'.

`model_report.highcharts.__init__.unicodeToHTMLEntities` (*text*)  
Converts unicode to HTML entities. For example '&' becomes '&amp;'.



## base

**class** `model_report.highcharts.base.CollectionObject`

Class to represent collection of dict values

**class** `model_report.highcharts.base.DictObject` (\*\*default)

Class to represent dict values

## options

`model_report.highcharts.options.get_highchart_data()`

Function helper that returns a basic all default values of highcharts javascript options.

## export\_pdf

## report

**class** `model_report.report.ReportAdmin` (*parent\_report=None, request=None*)

Class to represent a Report.

**base\_template\_name** = 'base.html'

Template file name to render the report.

**chart\_types** = ()

List of highchart types.

**check\_permissions** (*request*)

Override this method to another one raising Forbidden exceptions if you want to limit the access to the report

**exports** = ('excel', 'pdf')

Alternative render report as "pdf" or "csv".

**extra\_fields** = {}

Dictionary of fields that are aggregated to the query. Format {field\_name: Field instance}

**fields** = []

List of fields or lookup fields for query results to be listed.

**get\_column\_names** (*ignore\_columns={}*)

Return the list of columns

**get\_query\_set** (*filter\_kwargs*)

Return the the queryset

**get\_title** ()

Return the report title

**group\_totals** = {}

Dictionary with field name as key and function to calculate their values.

**inlines** = []

List of other's Report related to the main report.

**list\_filter** = ()

List of fields or lookup fields to filter data.

**list\_filter\_queryset** = {}

ForeignKey custom queryset

**list\_filter\_widget = {}**

Widget for list filter field

**list\_group\_by = ()**

List of fields or lookup fields to group data.

**list\_order\_by = ()**

List of fields or lookup fields to order data.

**list\_serie\_fields = ()**

List of fields to group by results in chart.

**model = None**

Primary django model to query.

**override\_field\_choices = {}**

#TODO

**override\_field\_filter\_values = {}**

#TODO

**override\_field\_formats = {}**

Dictionary with field name as key and function to parse their value after `override_field_values()`.

**override\_field\_labels = {}**

Dictionary with field name as key and function to parse the column label.

**override\_field\_values = {}**

Dictionary with field name as key and function to parse their original values.

```

override_field_values = {
    'men': men_format,
    'women': women_format
}

```

**override\_group\_value = {}**

#TODO

**query\_set = None**

#TODO

**report\_totals = {}**

Dictionary with field name as key and function to calculate their values.

**template\_name = 'model\_report/report.html'**

Template file name to render the report.

**title = None**

Title of the report.

**type = 'report'**

“report” for only report and “chart” for report and chart graphic results.

**class model\_report.report.ReportClassManager**

Class to handle registered reports class.

`model_report.report.autodiscover` (*module\_name='reports.py'*)

Auto-discover INSTALLED\_APPS report.py modules and fail silently when not present. Borrowed from `django.contrib.admin`

`model_report.report.cache_return` (*fun*)

Usages of this decorator have been removed from the ReportAdmin base class.

Caching method returns gets in the way of customization at the implementation level now that report instances can be modified based on request data.

`model_report.report.get_model_name(model)`

Returns string name for the given model

`model_report.report.is_date_field(field)`

Returns True if field is DateField or DateTimeField, otherwise False

## utils

**class** `model_report.utils.ReportRow`

Class to represent report rows

Attributes:

- `is_total` - defined as True if row is a group total row or report total row
- `is_caption` - TODO

**get\_css\_class()**

Render css classes to this row into html representation

**is\_value()**

Evaluate True if the row is a normal row or not

**class** `model_report.utils.ReportValue(value)`

Class to represent cells values for report rows

Attributes:

- `value` - real value from database
- `is_report_total` - defined as True if value is for showing in report total row
- `is_group_total` - defined as True if value is for showing in group total row
- `is_value` - defined as True if value is for showing in normal row

**format(value, instance)**

Format the value instance.

**text()**

Render as text the value. This function also format the value.

`model_report.utils.avg_column(values)`

Average values for any column

`model_report.utils.base_label(report, field)`

Basic label

`model_report.utils.count_column(values)`

Count values for any column

`model_report.utils.date_format(value, instance)`

Format cell value to friendly date string

`model_report.utils.round_format(value, instance)`

Format value to render with 2 decimal places

`model_report.utils.sum_column(values)`

Sum values for any column

`model_report.utils.usd_format` (*value, instance*)

Format cell value to money

`model_report.utils.yesno_format` (*value, instance*)

Format boolean value to render Yes or No if True or False

## widgets

**class** `model_report.widgets.RangeField` (*field\_class*, *widget=<class 'django.forms.widgets.TextInput'>*, \*args, \*\*kwargs)

Field to filter date values by range.

**class** `model_report.widgets.RangeWidget` (*widget, \*args, \*\*kwargs*)

Render 2 inputs with vDatepicker class in order to select a date range.

**m**

`model_report.highcharts.__init__`, 4  
`model_report.highcharts.base`, 5  
`model_report.highcharts.options`, 5  
`model_report.report`, 5  
`model_report.utils`, 7  
`model_report.widgets`, 8



**A**

autodiscover() (in module model\_report.report), 6  
 avg\_column() (in module model\_report.utils), 7

**B**

base\_label() (in module model\_report.utils), 7  
 base\_template\_name (model\_report.report.ReportAdmin attribute), 5

**C**

cache\_return() (in module model\_report.report), 6  
 chart\_types (model\_report.report.ReportAdmin attribute), 5  
 check\_permissions() (model\_report.report.ReportAdmin method), 5  
 CollectionObject (class in model\_report.highcharts.base), 5  
 count\_column() (in module model\_report.utils), 7

**D**

date\_format() (in module model\_report.utils), 7  
 DictObject (class in model\_report.highcharts.base), 5

**E**

exports (model\_report.report.ReportAdmin attribute), 5  
 extra\_fields (model\_report.report.ReportAdmin attribute), 5

**F**

fields (model\_report.report.ReportAdmin attribute), 5  
 format() (model\_report.utils.ReportValue method), 7

**G**

get\_column\_names() (model\_report.report.ReportAdmin method), 5  
 get\_css\_class() (model\_report.utils.ReportRow method), 7  
 get\_highchart\_data() (in module model\_report.highcharts.options), 5

get\_model\_name() (in module model\_report.report), 7  
 get\_query\_set() (model\_report.report.ReportAdmin method), 5  
 get\_title() (model\_report.report.ReportAdmin method), 5  
 group\_totals (model\_report.report.ReportAdmin attribute), 5

**H**

HTMLEntitiesToUnicode() (in module model\_report.highcharts.\_\_init\_\_), 4

**I**

inlines (model\_report.report.ReportAdmin attribute), 5  
 is\_date\_field() (in module model\_report.report), 7  
 is\_value() (model\_report.utils.ReportRow method), 7

**L**

list\_filter (model\_report.report.ReportAdmin attribute), 5  
 list\_filter\_queryset (model\_report.report.ReportAdmin attribute), 5  
 list\_filter\_widget (model\_report.report.ReportAdmin attribute), 5  
 list\_group\_by (model\_report.report.ReportAdmin attribute), 6  
 list\_order\_by (model\_report.report.ReportAdmin attribute), 6  
 list\_serie\_fields (model\_report.report.ReportAdmin attribute), 6

**M**

model (model\_report.report.ReportAdmin attribute), 6  
 model\_report.highcharts.\_\_init\_\_ (module), 4  
 model\_report.highcharts.base (module), 5  
 model\_report.highcharts.options (module), 5  
 model\_report.report (module), 5  
 model\_report.utils (module), 7  
 model\_report.widgets (module), 8

## O

`override_field_choices` (model\_report.report.ReportAdmin attribute), 6

`override_field_filter_values` (model\_report.report.ReportAdmin attribute), 6

`override_field_formats` (model\_report.report.ReportAdmin attribute), 6

`override_field_labels` (model\_report.report.ReportAdmin attribute), 6

`override_field_values` (model\_report.report.ReportAdmin attribute), 6

`override_group_value` (model\_report.report.ReportAdmin attribute), 6

## Q

`query_set` (model\_report.report.ReportAdmin attribute), 6

## R

`RangeField` (class in model\_report.widgets), 8

`RangeWidget` (class in model\_report.widgets), 8

`report_totals` (model\_report.report.ReportAdmin attribute), 6

`ReportAdmin` (class in model\_report.report), 5

`ReportClassManager` (class in model\_report.report), 6

`ReportRow` (class in model\_report.utils), 7

`ReportValue` (class in model\_report.utils), 7

`round_format()` (in module model\_report.utils), 7

## S

`sum_column()` (in module model\_report.utils), 7

## T

`template_name` (model\_report.report.ReportAdmin attribute), 6

`text()` (model\_report.utils.ReportValue method), 7

`title` (model\_report.report.ReportAdmin attribute), 6

`type` (model\_report.report.ReportAdmin attribute), 6

## U

`unicodeToHTMLEntities()` (in module model\_report.highcharts.\_\_init\_\_), 4

`usd_format()` (in module model\_report.utils), 7

## Y

`yesno_format()` (in module model\_report.utils), 8