django-model-report Documentation

Release 0.2.1

juanpex

Jun 28, 2017

Contents

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

CHAPTER 1

Demo

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 '&' becomes '&'.

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

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 form 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.

Python Module Index

m

Index

O

Q

R

S

T

U

Y