Django-Chartit Documentation

Release 0.1

Praveen Gollakota

Oct 25, 2017

Contents

Django-Chartit 2 can be used to plot the data on web pages from various models. The charts are rendered using the `Highcharts` and `jQuery` JavaScript libraries. Data in your database can be plotted as simple line charts, column charts, area charts, scatter plots, and many more chart types. Data can also be plotted as Pivot Charts where the data is grouped and/or pivoted by specific column(s).

CHAPTER 1

Features

- Can plot charts from models.
- Can plot data from multiple models on the same axis on a chart.
- Can plot pivot charts from models. Data can be pivoted by multiple columns.
- Can legend pivot charts by multiple columns.
- Can combine data from multiple models to plot on same pivot charts.
- Can plot a pareto chart, paretoed by a specific column.
- Can plot only a top few items per category in a pivot chart.

CHAPTER 2

Installation

You can install Django-Chartit 2 from PyPI. Just do

```
$ pip install django_chartit
```

You also need supporting JavaScript libraries. See the *Required JavaScript Libraries* section for more details.

CHAPTER 3

How to Use

Plotting a chart or pivot chart on a webpage involves the following steps.

1. Create a `DataPool` or `PivotDataPool` object that specifies what data you need to retrieve and from where.
2. Create a `Chart` or `PivotChart` object to plot the data in the `DataPool` or `PivotDataPool` respectively.
3. Return the `Chart`/`PivotChart` object from a django `view` function to the django template.
4. Use the `load_charts` template tag to load the charts to HTML tags with specific *ids*.

It is easier to explain the steps above with examples. So read on.

CHAPTER 4

How to Create Charts

Here is a short example of how to create a line chart. Let's say we have a simple model with 3 fields - one for month and two for temperatures of Boston and Houston.

```
class MonthlyWeatherByCity(models.Model):
    month = models.IntegerField()
    boston_temp = models.DecimalField(max_digits=5, decimal_places=1)
    houston_temp = models.DecimalField(max_digits=5, decimal_places=1)
```

And let's say we want to create a simple line chart of month on the x-axis and the temperatures of the two cities on the y-axis.

```
from chartit import DataPool, Chart

def weather_chart_view(request):
    #Step 1: Create a DataPool with the data we want to retrieve.
    weatherdata = \
        DataPool(
           series=
            [{'options': {
               'source': MonthlyWeatherByCity.objects.all()},
              'terms': [
                'month',
                'houston_temp',
                'boston_temp']}
             ])

    #Step 2: Create the Chart object
    cht = Chart(
            datasource = weatherdata,
            series_options =
              [{'options':{
                  'type': 'line',
                  'stacking': False},
                'terms':{
                  'month': [
```

```
                    'boston_temp',
                    'houston_temp']
                  }}],
            chart_options =
              {'title': {
                   'text': 'Weather Data of Boston and Houston'},
               'xAxis': {
                    'title': {
                       'text': 'Month number'}}})

    #Step 3: Send the chart object to the template.
    return render_to_response({'weatherchart': cht})
```

And you can use the `load_charts` filter in the django template to render the chart.

```
<head>
    <!-- code to include the highcharts and jQuery libraries goes here -->
    <!-- load_charts filter takes a comma-separated list of id's where -->
    <!-- the charts need to be rendered to                             -->
    {% load chartit %}
    {{ weatherchart|load_charts:"container" }}
</head>
<body>
    <div id='container'> Chart will be rendered here </div>
</body>
```

CHAPTER 5

How to Create Pivot Charts

Here is an example of how to create a pivot chart. Let's say we have the following model.

```
class DailyWeather(models.Model):
    month = models.IntegerField()
    day = models.IntegerField()
    temperature = models.DecimalField(max_digits=5, decimal_places=1)
    rainfall = models.DecimalField(max_digits=5, decimal_places=1)
    city = models.CharField(max_length=50)
    state = models.CharField(max_length=2)
```

We want to plot a pivot chart of month (along the x-axis) versus the average rainfall (along the y-axis) of the top 3 cities with highest average rainfall in each month.

```
from chartit import PivotDataPool, PivotChart

def rainfall_pivot_chart_view(request):
    #Step 1: Create a PivotDataPool with the data we want to retrieve.
    rainpivotdata = \
        PivotDataPool(
           series =
            [{'options': {
               'source': DailyWeather.objects.all(),
               'categories': ['month']},
              'terms': {
                'avg_rain': Avg('rainfall'),
                'legend_by': ['city'],
                'top_n_per_cat': 3}}
             ])

    #Step 2: Create the PivotChart object
    rainpivcht = \
        PivotChart(
            datasource = rainpivotdata,
            series_options =
              [{'options':{
```

```
                  'type': 'column',
                  'stacking': True},
                'terms':[
                  'avg_rain']}],
            chart_options =
              {'title': {
                   'text': 'Rain by Month in top 3 cities'},
               'xAxis': {
                    'title': {
                       'text': 'Month'}}})

    #Step 3: Send the PivotChart object to the template.
    return render_to_response({'rainpivchart': rainpivcht})
```

And you can use the `load_charts` filter in the django template to render the chart.

```
<head>
    <!-- code to include the highcharts and jQuery libraries goes here -->
    <!-- load_charts filter takes a comma-separated list of id's where -->
    <!-- the charts need to be rendered to                             -->
    {% load chartit %}
    {{ rainpivchart|load_charts:"container" }}
</head>
<body>
    <div id='container'> Chart will be rendered here </div>
</body>
```

CHAPTER 6

Rendering multiple charts

It is possible to render multiple charts in the same template. The first argument to `load_charts` is the Chart object or a list of Chart objects, and the second is a comma separated list of HTML IDs where the charts will be rendered.

When calling Django's `render` you have to pass all you charts as a list:

```
return render(request, 'index.html',
              {
                 'chart_list' : [chart_1, chart_2],
              }
        )
```

Then in your template you have to use the proper syntax:

```
<head>
    {% load chartit %}
    {{ chart_list|load_charts:"chart_1,chart_2" }}
</head>
<body>
    <div id="chart_1">First chart will be rendered here</div>
    <div id="chart_2">Second chart will be rendered here</div>
</body>
```

CHAPTER 7

Demo

The above examples are just a brief taste of what you can do with Django-Chartit 2. For more examples and to look at the charts in actions, see the demo website.

CHAPTER 8

API Reference

API Reference

How to retrieve data

DataPool

PivotDataPool

How to create the charts

Chart

PivotChart

How to use chartit django template filters

Quick Reference for `series` and `series_options`

PivotDataPool series	PivotChart series_options

`[{'options': {`
`  'source': SomeModel.objects.all(),`
`  'top_n_per_cat': 10, ...`
`  }`
`  'terms': {`
`  'any_name_here': Sum('a_valid_field`
`↪'),`
`  'some_other_name':{`
`  'func': Avg('a_valid_field),`
`  #any options to override`
`  ...`
`  },`
`  ...`
`  }`
`},`
`... #repeat dicts with 'options' &`
`↪'terms'`
`]`
	`[{'options': {`
	`  #any items from HighChart series.␣`
	`↪For ex.`
	`  'type': 'column'`
	`  },`
	`  'terms': [`
	`  'a_valid_term',`
	`  'other_valid_term': {`
	`  #any options to override. For ex.`
	`  'type': 'area',`
	`  ...`
	`  },`
	`  ...`
	`  ]`
	`  },`
	`  ... #repeat dicts with 'options' &`
	`↪'terms'`
	`  ]`

DataPool series	Chart series_options

`[{'options': {`
`  'source': SomeModel.objects.all(),`
 	
`  }`
`  'terms': [`
`  'a_valid_field_name',`
`  ..., # more valid field names`
`  {'any_name': 'a_valid_field_name',`
`  ... # more name:field_name pairs`
`  },`
`  ]`
`},`
`... #repeat dicts with 'options' &`
`↪'terms'`
`]`
	`[{'options': {`
	`  #any items from HighChart series.␣`
	`↪For ex.`
	`  'type': 'column'`
	`  },`
	`  'terms': {`
	`  'x_name': ['y_name', 'y_name', ...],`
	`  # only corresponding keys from␣`
	`↪DataPool`
	`  # terms are valid names.`
	`  ...`
	
	`  }`
	`  },`
	`  ... #repeat dicts with 'options' &`
	`↪'terms'`
	`  ]`

CHAPTER 9

Required JavaScript Libraries

The following JavaScript Libraries are required for using Django-Chartit 2.

- jQuery
- Highcharts

Note: While `Django-Chartit 2` itself is licensed under the BSD license, `Highcharts` is licensed under the Highcharts license and `jQuery` is licensed under both MIT License and GNU General Public License (GPL) Version 2. It is your own responsibility to abide by respective licenses when downloading and using the supporting JavaScript libraries.

CHAPTER 10

License