django-measurement Documentation

Release 1.0

Adam Coddington

Oct 08, 2017

Contents

A simple Django app providing for a simple way of using and storing weights and measures.

Contents:

CHAPTER 1

Using Measurement Objects in Forms

This is an example for a simple form field usage:

```
from django import forms
from django_measurement.forms import MeasurementField

class BeerForm(forms.Form):

    volume = MeasurementField(Volume)
```

You can limit the units in the select field by using the 'unit_choices' keyword argument. To limit the value choices of the MeasurementField uses the regular 'choices' keyword argument:

```
class BeerForm(forms.Form):

    volume = MeasurementField(
        measurement=Volume,
        unit_choices=(("l","l"), ("oz","oz")),
        choices=((1.0, 'one'), (2.0, 'two'))
    )
```

If unicode symbols are needed in the labels for a MeasurementField, define a LABELS dictionary for your subclassed MeasureBase object:

```
# -*- coding: utf-8 -*-
from sympy import S, Symbol

class Temperature(MeasureBase):
    SU = Symbol('kelvin')
    STANDARD_UNIT = 'k'
    UNITS = {
        'c': SU - S(273.15),
        'f': (SU - S(273.15)) * S('9/5') + 32,
        'k': 1.0
    }
    LABELS = {
```

```
        'c':u'°C',
        'f':u'°F',
        'k':u'°K',
    }
```

For a *MeasurementField* that represents a *BidimensionalMeasure*, you can set the separator either in settings.py (*MEASUREMENT_BIDIMENSIONAL_SEPARATOR is '/'* by default, add setting to override for all BiDimensionalMeasure subclasses) or override for an individual field with the kwarg bidimensional_separator:

```
speed = MeasurementField(
    measurement=Speed,
    bidimensional_separator=' per '
)

# Rendered option labels will now be in the format "ft per s", "m per hr", etc
```

CHAPTER 2

Installation

You can either install from pip:

```
pip install django-measurement
```

or checkout and install the source from the github repository:

```
git clone https://github.com/coddingtonbear/django-measurement.git
cd django-measurement
python setup.py install
```

CHAPTER 3

Measures

See python-measurement's documentation for information about what measures are available.

CHAPTER 4

Settings

MEASUREMENT_BIDIMENSIONAL_SEPARATOR

For any BidimensionalMeasure, what is placed between the primary and reference dimensions on rendered label

    MEASUREMENT_BIDIMENSIONAL_SEPARATOR = ” per “

Defaults to “/”. Can be overriden as kwarg *bidimensional_separator* for a given MeasurementField.

CHAPTER 5

Storing Measurement Objects

Suppose you were trying to cut back on drinking, and needed to store a log of how much beer you drink day-to-day; you might (naively) create a model like such:

```
from measurement.measures import Volume
from django_measurement.fields import MeasurementField
from django.db import models

class BeerConsumptionLogEntry(models.Model):
    name = models.CharField(max_length=255)
    volume = MeasurementField(Volume)

    def __str__(self):
        return '%s of %s' % (self.name, self.volume)
```

and assume you had a pint of Ninkasi's Total Domination; you'd add it to your log like so:

```
from measurement.measures import Volume

beer = BeerConsumptionLogEntry()
beer.name = 'Total Domination'
beer.volume = Volume(us_pint=1)
beer.save()

print beer # '1 us_pint of Total Domination'
```

Perhaps you next recklessly dove into your stash of terrible, but nostalgia-inducing Russian beer and had a half-liter of Baltika's #9; you'd add it to your log like so:

```
another_beer = BeerConsumptionLogEntry()
another_beer.name = '#9'
another_beer.volume = Volume(l=0.5)
another_beer.save()

print beer # '0.5 l of #9'
```

Note that although the original unit specified is stored for display, that the unit is abstracted to the measure's standard unit for storage and comparison:

```
print beer.volume                       # '1 us_pint'
print another_beer.volume               # '0.5 l'
print beer.volume > another_beer.volume # False
```

How is this data stored?

Since django-measurement v2.0 there value will be stored in a single float field.

CHAPTER 6

Using Measurement Objects

You can import any of the above measures from *measurement.measures* and use it for easily handling measurements like so:

```
from measurement.measures import Weight

w = Weight(lb=135) # Represents 135lbs
print w            # '135.0 lb'
print w.kg         # '61.234919999999995'
```

See Python-measurement's documentation for more information about interacting with measurements.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search