By **Vitor Freitas**

I'm a passionate software developer and researcher from Brazil, currently living in Finland. I write about Python, Django and Web Development on a weekly basis. Read more.

TUTORIAL

How to Implement Grouped Model Choice Field

Jan 2, 2019 6 minutes read 4 comments 11,999 views

(Picture: https://www.pexels.com/photo/apple-coffee-computer-cup-459653/)

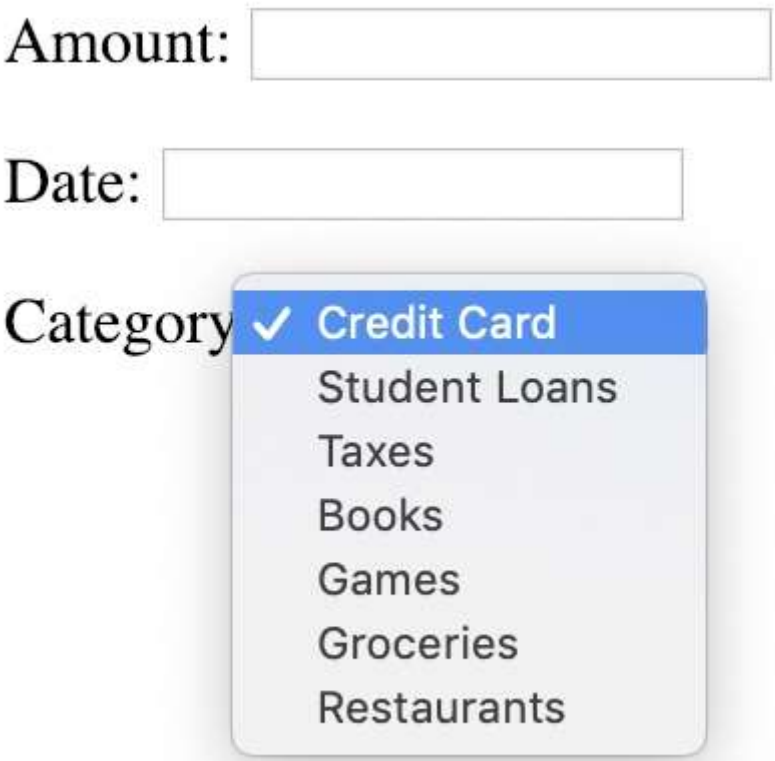
The Django forms API have two field types to work with multiple options: `ChoiceField` and `ModelChoiceField`.

Both use select input as the default widget and they work in a similar way, except that `ModelChoiceField` is designed to handle QuerySets and work with foreign key relationships.

A basic implementation using a `ChoiceField` would be:

```
class ExpenseForm(forms.Form):
    CHOICES = (
        (11, 'Credit Card'),
        (12, 'Student Loans'),
        (13, 'Taxes'),
        (21, 'Books'),
        (22, 'Games'),
        (31, 'Groceries'),
        (32, 'Restaurants'),
    )
    amount = forms.DecimalField()
```

```
    date = forms.DateField()
    category = forms.ChoiceField(choices=CHOICES)
```

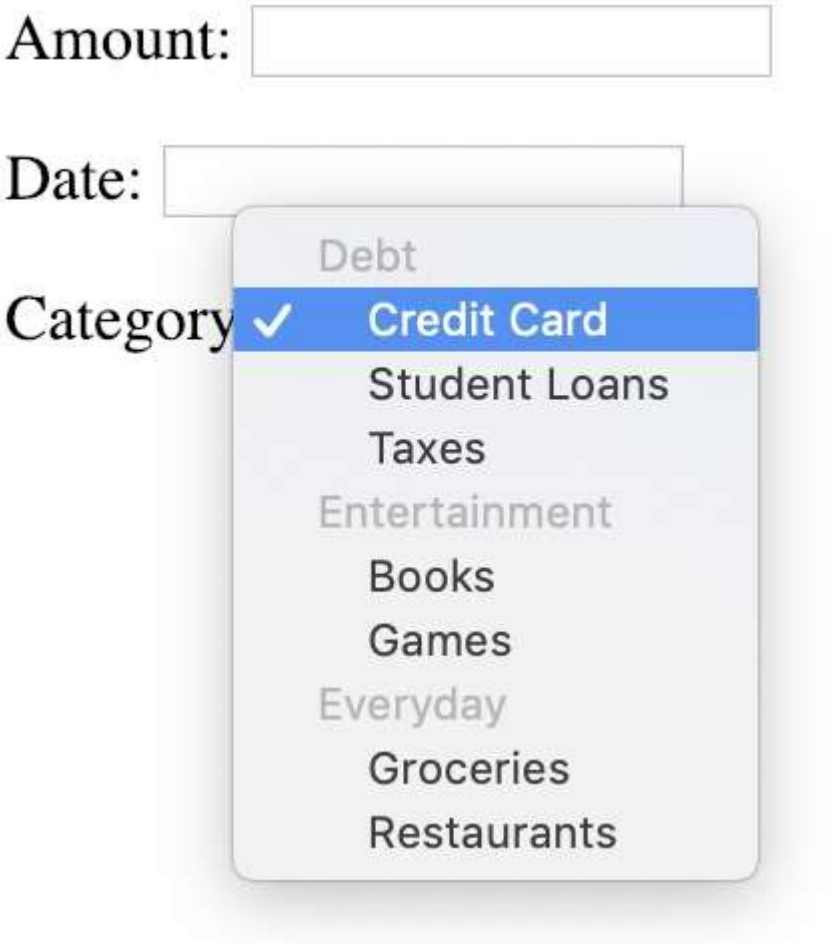


Grouped Choice Field

You can also organize the choices in groups to generate the `<optgroup>` tags like this:

```
class ExpenseForm(forms.Form):
    CHOICES = (
        ('Debt', (
```

```
            (11, 'Credit Card'),
            (12, 'Student Loans'),
            (13, 'Taxes'),
        )),
        ('Entertainment', (
            (21, 'Books'),
            (22, 'Games'),
        )),
        ('Everyday', (
            (31, 'Groceries'),
            (32, 'Restaurants'),
        )),
    )
    amount = forms.DecimalField()
    date = forms.DateField()
    category = forms.ChoiceField(choices=CHOICES)
```



Grouped Model Choice Field

When you are using a `ModelChoiceField` unfortunately there is no built-in solution.

Recently I found a nice solution on Django's ticket tracker, where someone proposed adding an `opt_group` argument to the `ModelChoiceField`.

While the discussion is still ongoing, Simon Charette proposed a really good solution.

Let's see how we can integrate it in our project.

First consider the following models:

models.py

```
from django.db import models

class Category(models.Model):
    name = models.CharField(max_length=30)
    parent = models.ForeignKey('Category', on_delete=models.CASCADE, null=

    def __str__(self):
        return self.name

class Expense(models.Model):
    amount = models.DecimalField(max_digits=10, decimal_places=2)
    date = models.DateField()
    category = models.ForeignKey(Category, on_delete=models.CASCADE)

    def __str__(self):
        return self.amount
```

So now our category instead of being a regular choices field it is now a model and the `Expense` model have a relationship with it using a foreign key.

If we create a `ModelForm` using this model, the result will be very similar to our first example.

To simulate a grouped categories you will need the code below. First create a new module named **fields.py**:

fields.py

```
from functools import partial
from itertools import groupby
from operator import attrgetter

from django.forms.models import ModelChoiceIterator, ModelChoiceField


class GroupedModelChoiceIterator(ModelChoiceIterator):
    def __init__(self, field, groupby):
        self.groupby = groupby
        super().__init__(field)

    def __iter__(self):
        if self.field.empty_label is not None:
            yield ("", self.field.empty_label)
        queryset = self.queryset
        # Can't use iterator() when queryset uses prefetch_related()
        if not queryset._prefetch_related_lookups:
            queryset = queryset.iterator()
        for group, objs in groupby(queryset, self.groupby):
            yield (group, [self.choice(obj) for obj in objs])


class GroupedModelChoiceField(ModelChoiceField):
    def __init__(self, *args, choices_groupby, **kwargs):
        if isinstance(choices_groupby, str):
            choices_groupby = attrgetter(choices_groupby)
        elif not callable(choices_groupby):
            raise TypeError('choices_groupby must either be a str or a cal
        self.iterator = partial(GroupedModelChoiceIterator, groupby=choice
        super().__init__(*args, **kwargs)
```

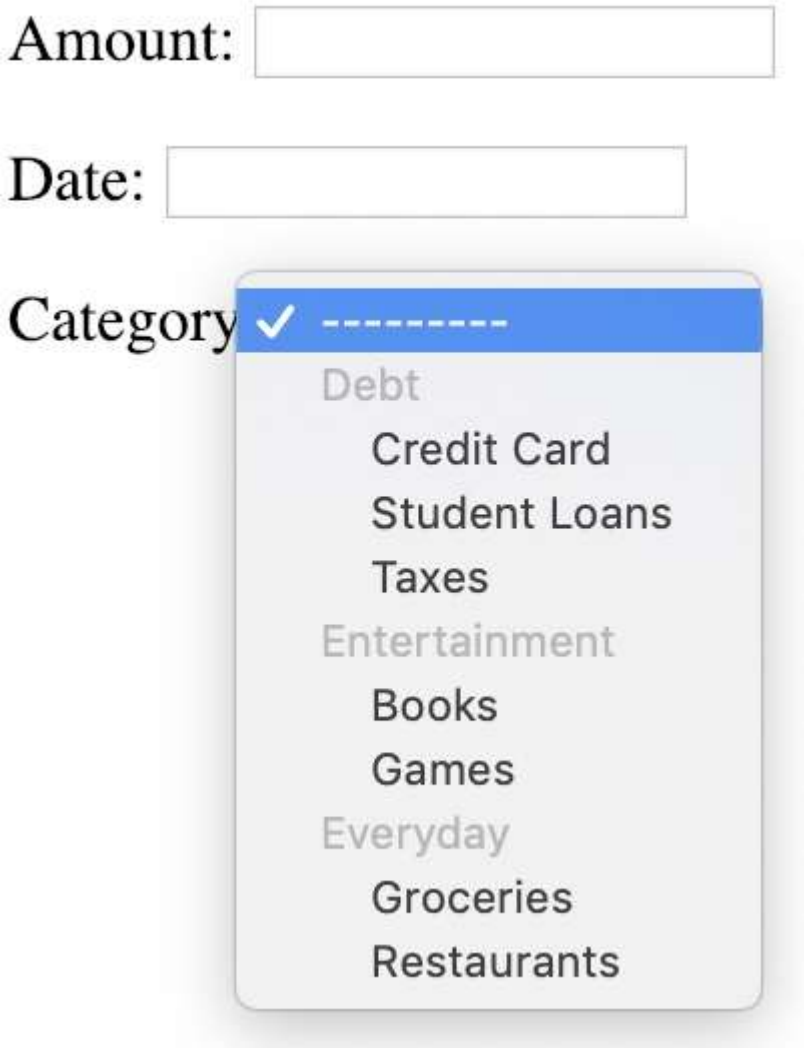
And here is how you use it in your forms:

forms.py

```python
from django import forms
from .fields import GroupedModelChoiceField
from .models import Category, Expense

class ExpenseForm(forms.ModelForm):
    category = GroupedModelChoiceField(
        queryset=Category.objects.exclude(parent=None),
        choices_groupby='parent'
    )

    class Meta:
        model = Expense
        fields = ('amount', 'date', 'category')
```



Because in the example above I used a self-referencing relationship I had to add the `exclude(parent=None)` to hide the "group categories" from showing up in the select input as a valid option.

Further Reading

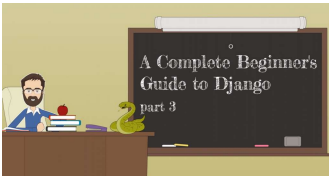
You can download the code used in this tutorial from GitHub: github.com/sibtc/django-grouped-choice-field-example

Credits to the solution Simon Charette on Django Ticket Track.

Related Posts

Advanced Form Rendering with Django Crispy Forms

How to Use Bootstrap 4 Forms With Django



A Complete Beginner's Guide to Django - Part 3

django forms

Share this post

Hey it's great to see this solution is being used in the wild :)

One thing I might suggest is that you either `select_related('parent')` or `prefetch_related('parent')` on the queryset you pass to `GroupedModelChoiceField` to avoid performing N queries when rendering the widget.

 • Reply • Share ›

Young Jun • 4 months ago

There is a bug which is not grouped together when I add several categories in admin page and back to home page.
Ex: I added category cat, cat001 (of cat), dog, dog001(of dog), dog002(of dog), cat002(of cat)
In form, when I try to select the category it looks like
cat
cat001
dog
dog001
dog002
cat
cat002

I think the 'groub_by' doesn't work.

 • Reply • Share ›

sunmar • 6 months ago

thx ... my problem diffrent sir ı wanna addtin my choiise fields and after write to diffrent html page.
ho caan ı do tihs

 • Reply • Share ›

Dan • 7 months ago

Elegant solution. Thanks!

 • Reply • Share ›

Launching our Community Forum

4 comments • 8 months ago

Avatar **Trung Bát** — Great guide, thanks for sharing.

How to Integrate Highcharts.js with Django

32 comments • a year ago

Avatar **Vitor Freitas** — Hi! You are free to translate and reproduce it on your website!Just remember to put a

A Complete Beginner's Guide to Django - Part 7

110 comments • 2 years ago

Akanemuse — Thank you Vitor, a

How to Implement Multiple User Types with Django

105 comments • 2 years ago

wanjohi kibui — A concern in the

Subscribe to our Mailing List

Receive updates from the Blog!

Your email address

SUBSCRIBE

Popular Posts

How to Extend Django User Model

How to Setup a SSL Certificate on Nginx for a Django Application

How to Deploy a Django Application to Digital Ocean

 // about contact faq cookies privacy policy