django-smart-selects documentation

Release 1.6.0.post3+g699426b

Jazzband

May 31, 2023

Contents

CHAPTER 1

Contents

1.1 Installation

1. Install `django-smart-selects` using a tool like `pip`:

```
$ pip install django-smart-selects
```

2. Add `smart_selects` to your `INSTALLED_APPS`

3. Add the `smart_selects` urls into your project's `urls.py`. This is needed for the `Chained Selects` and `Chained ManyToMany Selects`. For example:

```
urlpatterns = patterns('',
    url(r'^admin/', include(admin.site.urls)),
    url(r'^chaining/', include('smart_selects.urls')),
)
```

4. You will also need to include jQuery in every page that includes a field from `smart_selects`, or set `JQUERY_URL = True` in your project's `settings.py`.

1.2 Settings

`JQUERY_URL` : jQuery 2.2.0 is loaded from Google's CDN if this is set to `True`. If you would prefer to use a different version put the full URL here. Set `JQUERY_URL = False` to disable loading jQuery altogether.

`USE_DJANGO_JQUERY` : By default, `smart_selects` loads jQuery from Google's CDN. However, it can use jQuery from Django's admin area. Set `USE_DJANGO_JQUERY = True` to enable this behaviour.

1.3 Chained Selects

Given the following model:

```
class Continent(models.Model):
    name = models.CharField(max_length=255)

class Country(models.Model):
    continent = models.ForeignKey(Continent)
    name = models.CharField(max_length=255)

class Location(models.Model):
    continent = models.ForeignKey(Continent)
    country = models.ForeignKey(Country)
    area = models.ForeignKey(Area)
    city = models.CharField(max_length=50)
    street = models.CharField(max_length=100)
```

Once you select a continent, if you want only the countries on that continent to be available, you can use a `ChainedForeignKey` on the `Location` model:

```
from smart_selects.db_fields import ChainedForeignKey

class Location(models.Model):
    continent = models.ForeignKey(Continent)
    country = ChainedForeignKey(
        Country,
        chained_field="continent",
        chained_model_field="continent",
        show_all=False,
        auto_choose=True,
        sort=True)
    area = ForeignKey(Area)
    city = models.CharField(max_length=50)
    street = models.CharField(max_length=100)
```

1.3.1 ChainedForeignKey options

chained_field (required)

The `chained_field` indicates the field on the same model that should be chained to. In the `Continent`, `Country`, `Location` example, `chained_field` is the name of the field `continent` in model `Location`.

```
class Location(models.Model)
    continent = models.ForeignKey(Continent)
```

chained_model_field (required)

The `chained_model_field` indicates the field of the chained model that corresponds to the model linked to by the `chained_field`. In the `Continent`, `Country`, `Location` example, `chained_model_field` is the name of field `continent` in Model `Country`.

```
class Country(models.Model):
    continent = models.ForeignKey(Continent)
```

show_all (optional)

`show_all` indicates if only the filtered results should be shown or if you also want to display the other results further down.

auto_choose (optional)

`auto_choose` indicates if auto select the choice when there is only one available choice.

sort (optional)

`sort` indicates if the result set should be sorted lexicographically or not. Disable if you want to use the `Model.ordering` option. Defaults to `True`.

1.4 Chained ManyToMany Selects

The `ChainedManyToManyField` works as you would expect:

```
from smart_selects.db_fields import ChainedManyToManyField

class Publication(models.Model):
    name = models.CharField(max_length=255)

class Writer(models.Model):
    name = models.CharField(max_length=255)
    publications = models.ManyToManyField('Publication', blank=True,
↪ null=True)

class Book(models.Model):
    publication = models.ForeignKey(Publication)
    writer = ChainedManyToManyField(
        Writer,
        chained_field="publication",
        chained_model_field="publications")
    name = models.CharField(max_length=255)
```

1.4.1 Using chained fields in the admin

Do **not** specify the field in the `ModelAdmin filter_horizontal` list. Instead, simply pass `horizontal=True` to the `ChainedManyToManyField`:

```
from smart_selects.db_fields import ChainedManyToManyField

class Publication(models.Model):
    name = models.CharField(max_length=255)

class Writer(models.Model):
    name = models.CharField(max_length=255)
    publications = models.ManyToManyField('Publication', blank=True,
↪ null=True)

class Book(models.Model):
    publication = models.ForeignKey(Publication)
    writer = ChainedManyToManyField(
        Writer,
        horizontal=True,
        verbose_name='writer',
        chained_field="publication",
        chained_model_field="publications")
    name = models.CharField(max_length=255)
```

1.4.2 ChainedManyToManyField options

`chained_field` (required)

The `chained_field` indicates the field on the same model that should be chained to. In the `Publication`, `Writer`, `Book` example, `chained_field` is the name of the field `publication` in model `Book`.

```
class Book(models.Model):
    publication = models.ForeignKey(Publication)
```

`chained_model_field` (required)

The `chained_model_field` indicates the field of the chained model that corresponds to the model linked to by the `chained_field`. In the `Publication`, `Writer`, `Book` example, `chained_model_field` is the name of field `publications` in `Writer` model.

```
class Writer(models.Model):
    publications = models.ManyToManyField('Publication', blank=True,
↪ null=True)
```

auto_choose (optional)

auto_choose indicates if auto select the choice when there is only one available choice.

horizontal (optional)

This option will mixin Django's FilteredSelectMultiple to work in the Django admin as you expect

1.5 Grouped Selects

If you have the following model:

```
class Country(models.Model):
    continent = models.ForeignKey(Continent)

class Location(models.Model):
    continent = models.ForeignKey(Continent)
    country = models.ForeignKey(Country)
```

And you want to group countries by their continent in the HTML select list, you can use a GroupedForeignKey:

```
from smart_selects.db_fields import GroupedForeignKey

class Location(models.Model):
    continent = models.ForeignKey(Continent)
    country = GroupedForeignKey(Country, "continent")
```

1.6 Usage In Templates

In templates, continue using Django forms or modelforms as usual. Just include {{ form.media.js }} within your form. This will automatically add the following Javascript files into your page:

```
<script src="/static/smart-selects/admin/js/chainedfk.js"></script>
<script src="/static/smart-selects/admin/js/bindfields.js"></script>
```

Here'a a basic usage example:

```
<form method="post">
    {% csrf_token %}
    {{ form.media.js }}
    {{ form.as_p }}
```

(continues on next page)

(continued from previous page)

```
    <input type="submit" value="Submit" />
</form>
```

CHAPTER 2

Indices and tables

- search