
Easy-select2 Documentation

Release 1.2.2

Lobanov Stanislav aka asyncee

Oct 05, 2018

Contents

1	Installation	3
2	Quickstart	5
3	Configuration	7
4	Usage	9
5	Reference	11
5.1	Widgets	11
5.2	Utils	11
6	Sampleproject	13
6.1	Installation	13
6.2	Configuration	13
6.3	Usage	14
7	Changelog	15
7.1	Version 1.5.5	15
7.2	Version 1.5.4	15
7.3	Version 1.5.3	15
7.4	Version 1.5.2	16
7.5	Version 1.5.1	16
7.6	Version 1.5.0	16
7.7	Version 1.4.0	16
7.8	Version 1.3.4	16
7.9	Version 1.3.3	17
7.10	Version 1.3.2	17
7.11	Version 1.3.1	17
7.12	Version 1.3	17
7.13	Version 1.2	17
7.14	Version 1.1	19
8	Introduction	21
8.1	How it looks	21
9	Indices and tables	23

Contents:

CHAPTER 1

Installation

1. install this package as usual, using `python setup.py install`, `pip install django-easy-select2` or download sources and install to your python path.
2. add `easy_select2` to `INSTALLED_APPS` in your `settings.py`
3. Use `python manage.py collectstatic` or manually copy `easy_select2`'s static directory to your project's static directory (if you serve your static with `nginx`, for example).
4. Modify your `admin.py`.
5. Check out admin in browser.

CHAPTER 2

Quickstart

In your admin.py:

```
from django.contrib import admin
from easy_select2 import select2_modelform
from polls.models import Poll

PollForm = select2_modelform(Poll, attrs={'width': '250px'})

class PollAdmin(admin.ModelAdmin):
    form = PollForm
```

That's all. All your choice widgets are select2 widgets 250px wide.

CHAPTER 3

Configuration

`django-easy-select2` bundles jQuery and Select2 static files. You can use them, or specify your own files to include in widget.

To use bundled static, just install an application.

To use your custom static files, you can specify next settings in your `settings.py`:

- `SELECT2_JS` - path to `select2.js` file. Specify path without static directory, because full URL will be interpolated using `static` function from `staticfiles` application. Default: `easy_select2/vendor/select2/select2.min.js`
- `SELECT2_CSS` - path to `select2.css` file. Default: `easy_select2/vendor/select2/select2.min.css`
- `SELECT2_USE_BUNDLED_JQUERY` - default is `True`. Set to `False` if your already have included custom jQuery.

There are *Select2* and *Select2Multiple* widget classes for choice fields.

You can use *Select2* and *Select2Multiple* on any form field, as usual django widget:

```
class Form(forms.Form):
    field = forms.ModelChoiceField(queryset=qs, widget=Select2())
```

or:

```
class Form(forms.Form):
    field = forms.ModelMultipleChoiceField(queryset=qs, widget=Select2Multiple(
        select2attrs={'width': 'auto'}
    ))
```

Select2 and *Select2Multiple* is simple classes build with *Select2Mixin*:

```
class Select2Multiple(Select2Mixin, forms.SelectMultiple):
    pass

class Select2(Select2Mixin, forms.Select):
    pass
```

Select2Mixin is a simple widget mixin with predefined *Media* class and custom render method, which applies *\$.fn.select2()* method on html input.

Warning: Since version 1.2.9 *select2attrs* should be of type *dict* or *AssertionError* will be raised.

If you want to use it with all form fields automatically, without specifying each field, you can create your *ModelForm* class with *Meta* class constructed by custom *Meta* factory:

```
from easy_select2 import select2_modelform_meta
```

(continues on next page)

(continued from previous page)

```
class SomeModelForm(forms.ModelForm):
    Meta = select2_modelform_meta(SomeModel)
```

`select2_modelform_meta()` is a simple factory, that produces a `Meta` class with `model` attribute set to specified model and `widgets` attribute set to dictionary, containing all selectable fields on model. Every selectable field will be converted from standard widget to `Select2` or `Select2Multiple` widget.

If you are lazy, you can use `ModelForm` factory to build ready-to-use `ModelForm` for model with `select2_modelform()`:

```
from easy_select2 import select2_modelform

MyModelForm = select2_modelform(MyModel)
```

is the same like:

```
class MyModelForm(forms.ModelForm):
    Meta = select2_modelform_meta(models.SomeModelForm)
```

You can also specify your base form class instead of default `forms.ModelForm`:

```
from easy_select2 import select2_modelform

MyModelForm = select2_modelform(MyModel, form_class=forms.ModelForm)
```

`MyModelForm` is an instance of `ModelForm` with `model` attribute set to `MyModel`, and appropriate `Meta` class.

There is also an `apply_select2()` function that dynamically creates new widget class mixed with `Select2Mixin`.

Usage, for example:

```
class SomeModelForm(admin.ModelAdmin):
    class Meta:
        widgets = {
            'field': apply_select2(forms.Select),
        }
```

So, `apply_select2(forms.Select)` will return new class, named `Select2Select`, mixed with `Select2Mixin`.

5.1 Widgets

class `easy_select2.widgets.Select2` (*select2attrs=None, *args, **kwargs*)
Implement single-valued select widget with Select2.

class `easy_select2.widgets.Select2Mixin` (*select2attrs=None, *args, **kwargs*)
This mixin provides a mechanism to construct custom widget class, that will be rendered using Select2 input.
Generally should be mixed with widgets that render select input.

get_options ()
Return dictionary of options to be used by Select2.

render (*name, value, attrs=None, **kwargs*)
Extend base class's *render* method by appending javascript inline text to html output.

render_js_code (*id_, *args, **kwargs*)
Render html container for Select2 widget with options.

render_select2_options_code (*options, id_*)
Render options for select2.

class `easy_select2.widgets.Select2Multiple` (*select2attrs=None, *args, **kwargs*)
Implement multiple select widget with Select2.

5.2 Utils

`easy_select2.utils.apply_select2` (*widget_cls*)
Dynamically create new widget class mixed with Select2Mixin.

Args: *widget_cls*: class of source widget.

Usage, for example:

```
class SomeModelForm(admin.ModelAdmin):
    class Meta:
        widgets = {
            'field': apply_select2(forms.Select),
        }
```

So, `apply_select2(forms.Select)` will return new class, named `Select2Select`.

```
easy_select2.utils.select2_modelform(model, attrs=None, form_class=<class
                                     'easy_select2.forms.FixedModelForm'>)
```

Return `ModelForm` class for model with select2 widgets.

Arguments: `attrs`: select2 widget attributes (width, for example) of type *dict*. `form_class`: modelform base class, *forms.ModelForm* by default.

```
SomeModelForm = select2_modelform(models.SomeModelBanner)
```

is the same like:

```
class SomeModelForm(forms.ModelForm):
    Meta = select2_modelform_meta(models.SomeModelForm)
```

```
easy_select2.utils.select2_modelform_meta(model, meta_fields=None, widgets=None, at-
                                         trs=None, **kwargs)
```

Return *Meta* class with Select2-enabled widgets for fields with choices (e.g. `ForeignKey`, `CharField`, etc) for use with `ModelForm`.

Arguments: `model` - a model class to create *Meta* class for. `meta_fields` - dictionary with *Meta* class fields, for example, `{'fields': ['id', 'name']}`

attrs - select2 widget attributes (width, for example), must be of type *dict*.

****kwargs** - will be merged with `meta_fields`.

Sample project useful for testing django applications and other utility needs.

6.1 Installation

```
git clone https://github.com/asyncce/django-easy-select2.git
cd django-easy-select2/sampleproject
./bootstrap.sh # creates virtualenv and installs django (from requirements.txt)
source env/bin/activate
./manage.py migrate
./manage.py createsuperuser
./manage.py runserver
```

6.2 Configuration

Project ships with sane defaults (settings are pretty verbose):

- sqlite3 database
- filebased cache in /tmp
- cached sessions
- console email backend
- non-cached template loaders
- *css/js/img* default static dirs
- default *templates* directory
- nice default loggers

6.3 Usage

After you bootstrapped a project, you can fill it with some data and play with Note model admin.

7.1 Version 1.5.5

- Fixed cascading issue-52 bug
- Introducing django's JQuery to easy-select2 for accessing django's JQuery

7.2 Version 1.5.4

- Work with jQuery instead of depending on \$, thanks to *leibowitz*
- Fixed a problem with django admin tabularinline (#52), thanks to *leibowitz*

7.3 Version 1.5.3

Warning: Version 1.5.3 changes, read below.

- Fixed #50 (partly remaining bug in v 1.5.2) - 1st record addition fixed for “Doesn't work for dynamically added forms in inline admins”
- updated `easy_select.js` to fail case when `easy-select2.js[line#65]$(e.target).find('div.field-easy-select2:not([id*="__prefix__"]')` is not able to find elements while in `DOMNodeInserted` corresponding to other node insertion than `select2` widgets.

7.4 Version 1.5.2

Warning: Version 1.5.2 changes, read below.

- Fixed #45 (remaining bug in v 1.5.0) - 1st record addition fixed for “Doesn’t work for dynamically added forms in inline admins”
- updated easy_select js to handle DomNodeInserted Event for select2() dynamic initialization for all instances

7.5 Version 1.5.1

Warning: Version 1.5.1 changes, read below.

- Fixed #45 - “Doesn’t work for dynamically added forms in inline admins”
- updated easy_select js to handle DomNodeInserted Event for select2() dynamic initialization

7.6 Version 1.5.0

Warning: Version 1.5.0 major changes, read below.

- Fixed #44 - “mark_safe problem”
- Support for select2 constructor argument injection, within separate initialization block with for select2.
- updated easy-select2 wrapper initialization. Updated to JQuery plugin code design for JS code injection, allowing direct injection of select2 constructor arguments.

7.7 Version 1.4.0

Warning: Version 1.4.0 introduced backward incompatible changes, read below.

- Fixed #38 - “Related model combobox doesn’t update after add in Django 2”
- Dropping support for Python 2.x
- Django 2.0+ support. Demoapp updated to reflect the needed changes.
- Python 3.x+ support (recommended Python3.4 and above)

7.8 Version 1.3.4

- Django 1.11 support

7.9 Version 1.3.3

- Fixed #29 — “Application breaks dumpdata command”

7.10 Version 1.3.2

- Fixed #24, big thanks to *Andrzej Mateja*

7.11 Version 1.3.1

- support for django staticfiles storage, thanks to *martolini* for idea

7.12 Version 1.3

Warning: Version 1.3 introduced backward incompatible changes, read below.

- *Select2* updated to 4.0.0
- updated *jQuery* to 2.1.3
- removed deprecated *select2_meta_factory*, *Select2TextMixin* and *Select2TextInput*.

7.13 Version 1.2

7.13.1 1.2.13

- fixed issue #22, thanks to *zeta83*

7.13.2 1.2.12

- fixed issue#2

7.13.3 1.2.11

- fixed issue#15 - “RemovedInDjango18Warning”

7.13.4 1.2.10

- fixed issue#14 - README.rst is not included in MANIFEST.in

7.13.5 1.2.9

- fixed issue#12 “Inline relations: “Add another <Model>” breaks dropdown boxes”

<p>Warning: Version 1.2.9 introduced backward incompatible change: <code>select2attrs</code> argument of <code>Select2Mixin.__init__</code> must be of type dict</p>

7.13.6 1.2.8

- fixed incorrect instructions in `help_text` of ManyToMany fields #2, thanks to *bashu*.

7.13.7 1.2.7

- `setup.py` fixes (issue #11), thanks to *JensTimmerman*.

7.13.8 1.2.6

- Extended `select2_modelform` function with `form_class` argument to specify form base class explicitly (issue #10).

7.13.9 1.2.5

- Fixed issue #9 “`apply_select2` not imported in `__init__`” thanks to *ocZio* for bug report.

7.13.10 1.2.4

- Fixed issue #6 “Select will not update selection after adding a new option”, thanks to *ismaelbej* for bug report.

7.13.11 1.2.3

- Python 3.3 support, thanks to *dzerrenner*

7.13.12 1.2.2

- Rendering `select2attrs` as unicode or json based on type

Now, if `select2attrs` is instance of `basestring` (`str` or `unicode`), it will be casted to `unicode`, else it will be turned to `json` string.

7.13.13 1.2.1

- Extended package-level imports with `Select2TextInput`

7.13.14 1.2.0

- added Select2TextInput, thanks to *mkoistinen*

7.14 Version 1.1

7.14.1 1.1.1

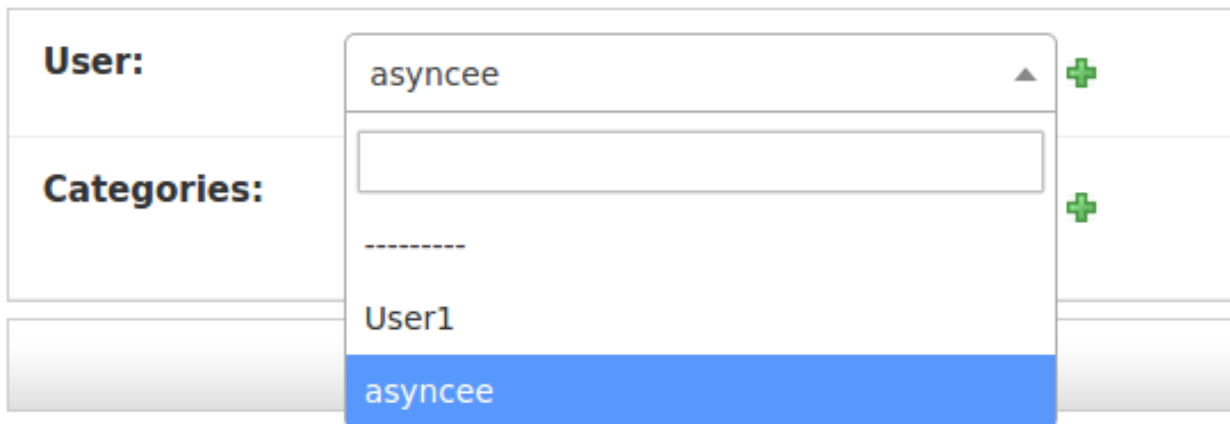
- issue#1 fix (django-admin-sortable compatibility), thanks to @mkoistinen

This is django application that brings select2 widget to select inputs in admin.

8.1 How it looks

Select one of existing values with single-valued choice field (ForeignKeyField, for example):


Add note




The screenshot shows the Django Admin interface for adding a note. The form has two main sections: 'User:' and 'Categories:'. The 'User:' field is a select2 widget with a dropdown menu open, showing the current selection 'asyncee' and a list of options including 'User1' and 'asyncee'. The 'Categories:' field is a multiple-valued choice field with a search bar and a list of options, also showing a green plus button for adding new categories.

Easily select 1 or more “categories” for your project, you can also add a new one in the normal, Django-Admin manner by using the green + button with multiple-valued choice field (ManyToManyField):

Add note

User: 

Categories: 

- Personal
- Project X
- Project Y

Continue to *Installation*.

CHAPTER 9

Indices and tables

- `genindex`
- `modindex`
- `search`

e

`easy_select2.utils`, [11](#)
`easy_select2.widgets`, [11](#)

A

apply_select2() (in module easy_select2.utils), 11

E

easy_select2.utils (module), 11

easy_select2.widgets (module), 11

G

get_options() (easy_select2.widgets.Select2Mixin method), 11

R

render() (easy_select2.widgets.Select2Mixin method), 11

render_js_code() (easy_select2.widgets.Select2Mixin method), 11

render_select2_options_code() (easy_select2.widgets.Select2Mixin method), 11

S

Select2 (class in easy_select2.widgets), 11

select2_modelform() (in module easy_select2.utils), 12

select2_modelform_meta() (in module easy_select2.utils), 12

Select2Mixin (class in easy_select2.widgets), 11

Select2Multiple (class in easy_select2.widgets), 11