django-smart-autoregister Documentation

Release 1.7.0

Paulo Cheque

Mar 09, 2017

Contents

Django-Smart-AutoRegister (DSA) is a Django tool that automatically configure the ModelAdmin classes of your application using some good patterns:

- The default behavior is to protect your application against reading without intention the entire data table because some foreign key relationship.
- It search by all pertinent fields.
- It create filter for all pertinent fields.

CHAPTER 1

Getting Started

Basic Example of Usage

In some `admin.py` file:

```
from django_smart_autoregister import auto_configure_admin_for_model
from django_smart_autoregister import auto_configure_admin
from django.contrib.auth.models import User

# ignore if User has already been registered
auto_configure_admin_for_model(User)

# replace User admin configuration if User model has already been registered
auto_configure_admin_for_model(User, override=True)

# or
auto_configure_admin()

# or
auto_configure_admin(exclude_applications=['django.contrib.auth'])

# or
auto_configure_admin(applications=['your_app1', 'your_app2'])
```

Installation

```
pip install django-smart-autoregister
```

or:

```
1. Download zip file
2. Extract it
3. Execute in the extracted directory: python setup.py install
```

Development version

```
pip install -e git+git@github.com:paulocheque/django-smart-autoregister.git
↪#egg=django-smart-autoregister
```

requirements.txt

```
django-smart-autoregister==<VERSION>
# or use the development version
git+git://github.com/paulocheque/django-smart-autoregister.git#egg=django-smart-
↪autoregister
```

Upgrade

```
pip install django-smart-autoregister --upgrade --no-deps
```

Compatibility

- Tested with Django 1.4, 1.5, 1.6, 1.7, 1.8
- Tested with Python 2.7, 3.3, 3.4 and PyPy

Motivation

It is boring to configure Django admin application for every model. It is a replicated task most of the time. Just for some special customization or special behavior that we need to waste some time to do this.

External references

- TODO

CHAPTER 2

Features and Configurations

https://docs.djangoproject.com/en/dev/ref/contrib/admin/

Registering your models

Do not use the standard Django method to register your application:

```
from django.contrib import admin
admin.site.register(YourModel, YourModelAdmin)
```

Rather that, register your models using one of the following functions, by *model* or *app* that it is explained in the following sections:

- **auto_configure_admin_for_model(model, override=False, **kwargs)**
- **auto_configure_admin_for_app(app, override=False)**
- **auto_configure_admin(applications=[], exclude_applications=[], override=False)**

Configuration per model

Uset the `auto_configure_admin_for_model` method:

```
from django_smart_autoregister import auto_configure_admin_for_model
from django.contrib.auth.models import User

auto_configure_admin_for_model(User)
```

To override the automatically generated config, you can set the ModelAdmin attribute as a `auto_configure_admin_for_model` method parameter:

```
auto_configure_admin_for_model(User, raw_id_fields=[], search_fields=['email',
↪'username'])
```

Pay attention you can receive a AlreadyRegistered exception if you are trying to configure a model that has already been registered. If you want to override previous configurations, you can use like this:

```
auto_configure_admin_for_model(User, raw_id_fields=[], search_fields=['email',
↪'username'], override=True)
```

Configure all models

You can configure automatically all models of a list of applications defined in `settings.INSTALLED_APPS`:

```
# admin.py
from django_smart_autoregister import auto_configure_admin
auto_configure_admin(['your_app1', 'your_app2'])

# or
auto_configure_admin(applications=['your_app1', 'your_app2'])
```

Or you can automatically configure the admin for all apps defined in `settings.INSTALLED_APPS`:

```
from django_smart_autoregister import auto_configure_admin
auto_configure_admin()
```

And to exclude some application:

```
from django_smart_autoregister import auto_configure_admin
auto_configure_admin(exclude_applications=['django.contrib.auth'])
```

Settings

DSA_FIELD_STRATEGY

Sometimes you can to override the default intelligence of the tool. To do that, define the **DSA_FIELD_STRATEGY** settings in your `settings.py` file:

```
# Functions that receive a field instance (e.g: `models.CharField(max_length=1)`) and␣
↪return a boolean
# If this functions returns True, the field will be included in the admin attribute␣
↪config (e.g: `Admin.raw_id_fields`)
DSA_FIELD_STRATEGY = {
    'raw_id_fields': lambda field: True,
}
```

For example, lets create a strategy that we will configure the *list_filter* for every model field that contains *choices*:

```
# This module offers a set of useful introspection functions to manipulate Django␣
↪models/fields
from django_smart_autoregister.django_helper import field_has_choices

class MyModel(models.Model):
    my_choices = models.CharField(max_length=2, choices=(('A', 'A'), ('B', 'B')))
    another_choices = models.CharField(max_length=2)

DSA_FIELD_STRATEGY = {
```

```
    'list_filter': lambda field: field_has_choices(field),
}

# `Admin.list_filter` will return `['my_choices']`
```

Or we can customize some default values:

```
DSA_FIELD_STRATEGY = {
    'list_per_page': 20,
    'list_max_show_all': 50,
}
```

DSA_FULL_STRATEGY

Sometimes you can put some intelligence after the value (e.g: *list_field = [field1, field2, field7]*) was created by the tool or by the **DSA_FIELD_STRATEGY** settings. To to that define a strategy that will receive the generated value and fixed it according to your ideas throw the settings **DSA_FULL_STRATEGY**:

```
# function(values) => values
DSA_FULL_STRATEGY = {
    'raw_id_fields': lambda values: values_updated,
}
```

For example, we want to show in the maximum 5 columns in the admin list table:

```
DSA_FULL_STRATEGY = {
    'list_display': lambda values: values[0:5],
}

# Without this config: Admin.list_display = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
# With this last config: Admin.list_display = ['a', 'b', 'c', 'd', 'e']
```

CHAPTER 3

About

Change Log

Date format: yyyy/mm/dd

Version 0.0.5 - 2017/03

- <http://pypi.python.org/pypi/django-smart-autoregister/0.0.5>
- [Update] Fixed test compability

Version 0.0.4 - 2017/03

- <http://pypi.python.org/pypi/django-smart-autoregister/0.0.4>
- [Update] Django 1.10 compatibility issues
- [Bugfix] Fixed internal issues for testing/Travis

Version 0.0.3 - 2015/05

- <http://pypi.python.org/pypi/django-smart-autoregister/0.0.3>
- [Update] No more deprecated methods

Version 0.0.2 - 2015/05

- <http://pypi.python.org/pypi/django-smart-autoregister/0.0.2>
- [New] Django 1.8 support

Version 0.0.1 - 2014/10/04

- <http://pypi.python.org/pypi/django-smart-autoregister/0.0.1>
- [New] Raw id fields for all Foreign Keys
- [New] List display for all string, date, number and boolean columns
- [New] List display links for all columns
- [New] List filter for all fields with choices and booleans
- [New] Search fields for all string columns
- [New] DSA_FIELD_STRATEGY settings
- [New] DSA_FULL_STRATEGY settings

Collaborators

Paulo Cheque <http://twitter.com/paulocheque> <https://github.com/paulocheque>

Pull Requests tips

About commit messages

- Messages in english only
- All messages have to follow the pattern: "[TAG] message"
- TAG have to be one of the following: new, update, bugfix, delete, refactoring, config, log, doc, mergefix

About the code

- One change (new feature, update, refactoring, bugfix etc) by commit
- All bugfix must have a test simulating the bug
- All commit must have 100% of test coverage

Running tests

Command:

```
python manage.py test --with-coverage --cover-inclusive --cover-html --cover-
↪package=your_package.*
```

TODO list

Tests and Bugfixes

-

Features

-

Documentation

-

CHAPTER 4

Indices and tables

- genindex
- modindex
- search