A PYTHONIC FULL-TEXT SEARCH

PAOLO MELCHIORRE ~ @pauloxnet

full text search

16 results for *full text search* in version 3.1

Getting Help

Language: **en**

Documentation version: **3.1**

Full text search

API Reference » contrib packages » django.contrib.postgres

Search

Paolo Melchiorre

CTO @ 20tab

- Remote worker
- Software engineer
- Python developer
- Django contributor

→| Pythonic

>>> import this

"**Beautiful** is better than *ugly*.

Explicit is better than *implicit*.

Simple is better than *complex*.

Complex is better than *complicated*."

— "The Zen of Python", *Tim Peters*

Full-text search

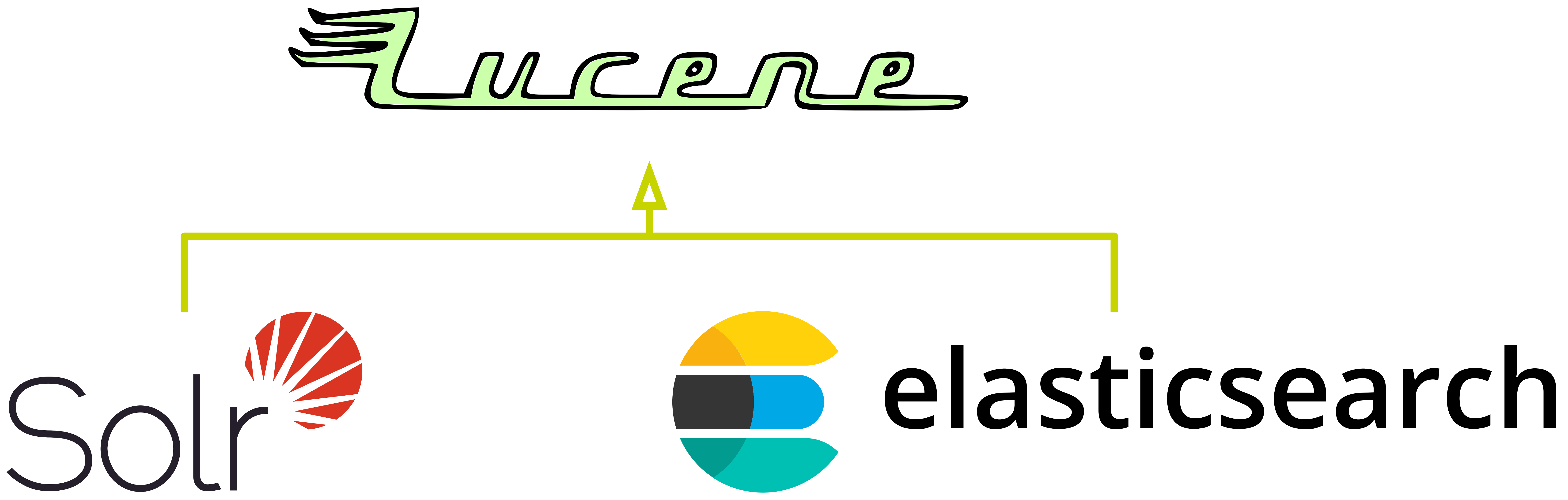
"… *techniques* for **searching**

… computer-stored **document** …

in a **full-text** *database*."

— "Full-text search", *Wikipedia*

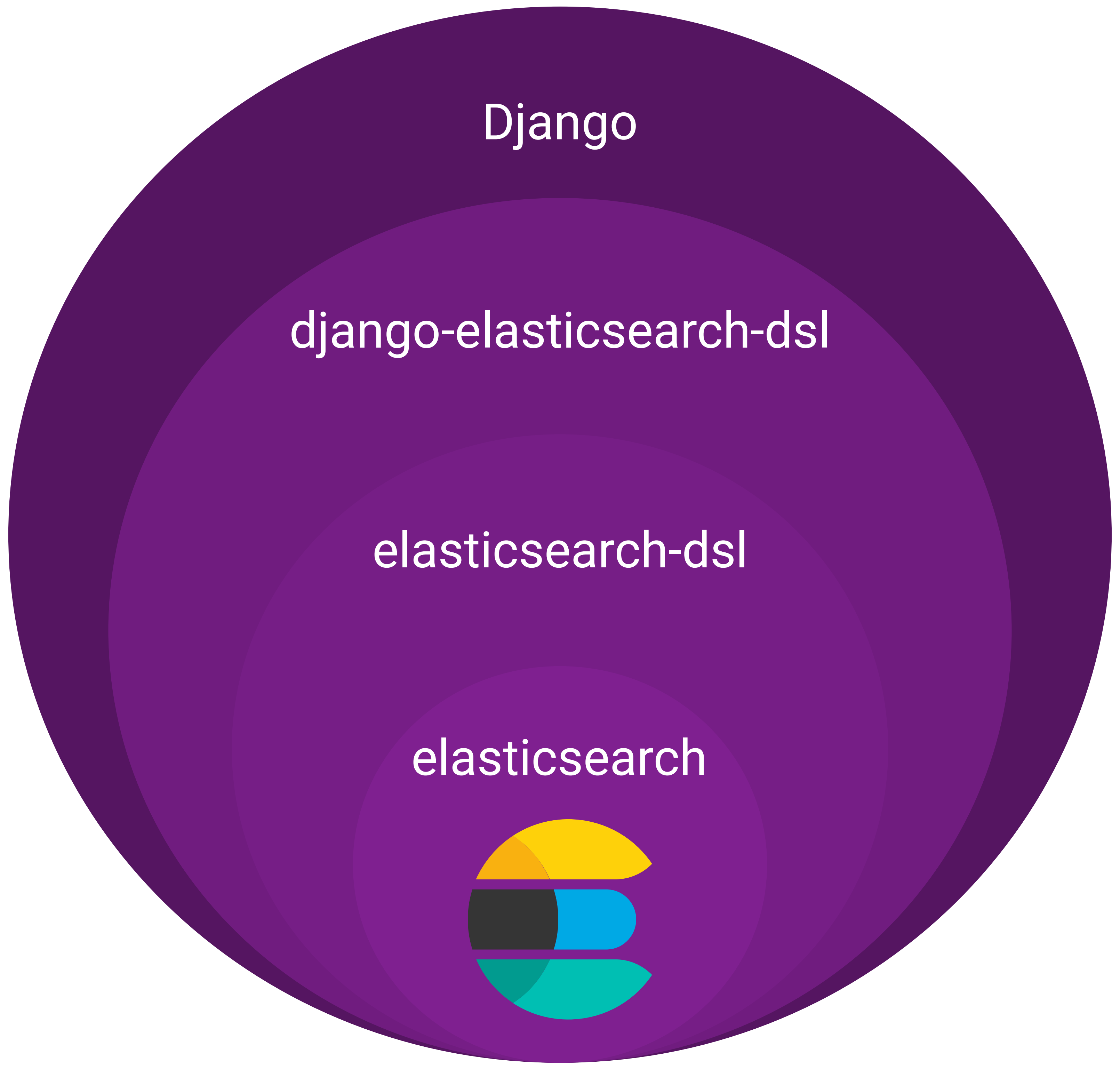
Popular engines



Documenti in evidenza

docs.italia.it

A “Read the Docs” fork



External engines

PROS	CONS
Popular	Driver
Full featured	Query language
Resources	Synchronization

Sorry!

This slide is no longer available.

PostgreSQL

- **Full text search** *(v8.3 ~2008)*
- Data type *(tsquery, tsvector)*
- Special indexes *(GIN, GiST)*
- Phrase search *(v9.6 ~2016)*
- JSON support *(v10 ~2017)*
- Web search *(v11 ~2018)*
- New languages *(v12 ~2019)*

Document

"... the **unit** of searching

in a full-text search **system**;

e.g., a magazine **article** ..."

— "Full Text Search", *PostgreSQL Documentation*

Django

- **Full text search** *(v1.10 ~2016)*
- django.contrib.postgres
- Fields, expressions, functions
- GIN index *(v1.11 ~2017)*
- GiST index *(v2.0 ~2018)*
- Phrase search *(v2.2 ~2019)*
- Web search *(v3.1 ~2020)*

→| Document-based search

- Weighting
- Categorization
- Highlighting
- Multiple languages

```python
"""Blogs models."""

from django.contrib.postgres import search
from django.db import models

class Blog(models.Model):
    name = models.CharField(max_length=100)
    tagline = models.TextField()

class Author(models.Model):
    name = models.CharField(max_length=200)

class Entry(models.Model):
    blog = models.ForeignKey(Blog, on_delete=models.CASCADE)
    headline = models.CharField(max_length=255)
    body_text = models.TextField()
    authors = models.ManyToManyField(Author)
    search_vector = search.SearchVectorField()
```

```
"""Field lookups."""

from blog.models import Author

Author.objects.filter(name__contains="Terry")
[<Author: Terry Gilliam>, <Author: Terry Jones>]

Author.objects.filter(name__icontains="ERRY")
[<Author: Terry Gilliam>, <Author: Terry Jones>, <Author: Jerry Lewis>]
```

```python
"""Unaccent extension."""

from django.contrib.postgres import operations
from django.db import migrations

class Migration(migrations.Migration):
    operations = [operations.UnaccentExtension()]


"""Unaccent lookup."""

from blog.models import Author

Author.objects.filter(name__unaccent="Helene Joy")
[<Author: Hélène Joy>]
```

```
"""Trigram extension."""

from django.contrib.postgres import operations
from django.db import migrations

class Migration(migrations.Migration):
    operations = [operations.TrigramExtension()]


"""Trigram similar lookup."""

from blog.models import Author

Author.objects.filter(name__trigram_similar="helena")
[<Author: Helen Mirren>, <Author: Helena Bonham Carter>]
```

```
"""App installation."""

INSTALLED_APPS = [
    # …
    "django.contrib.postgres",
]


"""Search lookup."""

from blog.models import Entry

Entry.objects.filter(body_text__search="cheeses")
[<Entry: Cheese on Toast recipes>, <Entry: Pizza Recipes>]
```

```python
"""SearchVector function."""

from django.contrib.postgres import search
from blog.models import Entry


SEARCH_VECTOR = search.SearchVector("body_text", "blog__name")



entries = Entry.objects.annotate(search=SEARCH_VECTOR)
entries.filter(search="cheeses")
[<Entry: Cheese on Toast recipes>, <Entry: Pizza Recipes>]
```

```python
"""SearchQuery expression."""

from django.contrib.postgres import search
from blog.models import Entry


SEARCH_VECTOR = search.SearchVector("body_text")
SEARCH_QUERY = search.SearchQuery("pizzas OR toasts", search_type="websearch")


entries = Entry.objects.annotate(search=SEARCH_VECTOR)
entries.filter(search=SEARCH_QUERY)
[<Entry: Cheese on Toast recipes>, <Entry: Pizza Recipes>]
```

```python
"""SearchConfig expression."""

from django.contrib.postgres import search
from blog.models import Entry


SEARCH_VECTOR = search.SearchVector("body_text", config="french")
SEARCH_QUERY = search.SearchQuery("œuf", config="french")


entries = Entry.objects.annotate(search=SEARCH_VECTOR)
entries.filter(search=SEARCH_QUERY)
[<Entry: Pain perdu>]
```

```python
"""SearchRank function."""

from django.contrib.postgres import search
from blog.models import Entry


SEARCH_VECTOR = search.SearchVector("body_text")
SEARCH_QUERY = search.SearchQuery("cheese OR meat", search_type="websearch")
SEARCH_RANK = search.SearchRank(SEARCH_VECTOR, SEARCH_QUERY)

entries = Entry.objects.annotate(rank=SEARCH_RANK)
entries.order_by("-rank").filter(rank__gt=0.01).values_list("headline", "rank")
[('Pizza Recipes', 0.06079271), ('Cheese on Toast recipes', 0.044488445)]
```

```python
"""SearchVector weight attribute."""

from django.contrib.postgres import search
from blog.models import Entry

SEARCH_VECTOR = search.SearchVector("headline", weight="A") \
              + search.SearchVector("body_text", weight="B")
SEARCH_QUERY = search.SearchQuery("cheese OR meat", search_type="websearch")
SEARCH_RANK = search.SearchRank(SEARCH_VECTOR, SEARCH_QUERY)

entries = Entry.objects.annotate(rank=SEARCH_RANK).order_by("-rank")
entries.values_list("headline", "rank")
[('Cheese on Toast recipes', 0.36), ('Pizza Recipes', 0.24), ('Pain perdu', 0)]
```

```python
"""SearchHeadline function."""

from django.contrib.postgres import search
from blog.models import Entry



SEARCH_QUERY = search.SearchQuery("pizzas OR toasts", search_type="websearch")
SEARCH_HEADLINE = search.SearchHeadline("headline", SEARCH_QUERY)

entries = Entry.objects.annotate(highlighted_headline=SEARCH_HEADLINE)
entries.values_list("highlighted_headline", flat=True)
['Cheese on <b>Toast</b> recipes', '<b>Pizza</b> Recipes', 'Pain perdu']
```

```python
"""SearchVector field."""

from django.contrib.postgres import search
from blog.models import Entry


SEARCH_VECTOR = search.SearchVector("body_text")
SEARCH_QUERY = search.SearchQuery("pizzas OR toasts", search_type="websearch")


Entry.objects.update(search_vector=SEARCH_VECTOR)
Entry.objects.filter(search_vector=SEARCH_QUERY)
[<Entry: Cheese on Toast recipes>, <Entry: Pizza Recipes>]
```

Django 1.10 release notes

August 1, 2016

What's new in Django 1.10

Full text search for PostgreSQL

`django.contrib.postgres` now includes a collection of databas
the full text search engine. You can search across multiple fields in yo
combine the searches with other lookups, use different language configurations and weightings,

An old search

- English-only search
- HTML tag in results
- Sphinx generation
- PostgreSQL database
- External search engine

Django developers feedback

CONS

Work to do

Features

Database workload

PROS

Maintenance

Light setup

Dogfooding

europython
Rimini
9-16 JULY 2017

Search or jump to...

Pull requests Issues Marketplace Explore

django / djangoproject.com

Sponsor | Unwatch 121 | Unstar 1.3k | Fork 644

Code | Issues 53 | Pull requests 11 | Actions | Security | Insights

Updated docs search to use PostgreSQL full-text search #797

Edit | Open with

Merged **timgraham** merged 1 commit into `django:master` from `pauloxnet:pg_fts` on Nov 22, 2017

Conversation 64 | Commits 1 | Checks 0 | Files changed 16 | +194 −312

pauloxnet commented on Nov 12, 2017 • edited by timgraham · Contributor

All Full-Text Search features based on Elasticsearch replaced with PostgreSQL FTS
https://groups.google.com/d/topic/django-developers/kxH56zaAeZY/discussion

👍 17

Reviewers
apollo13
timgraham

Assignees
No one assigned

→| djangoproject.com

Full-text search features

- Multilingual
- PostgreSQL based
- Clean results
- Low maintenance
- Easier to setup

What’s next

- Misspelling support
- Search suggestions
- Highlighted results
- Web search syntax
- Search statistics

→| Tips

- docs in djangoproject.com
- details in postgresql.org
- source code in github.com
- questions in stackoverflow.com

→| License

CC BY-SA 4.0

DO MORE WITH LESS

20tab.com

info@20tab.com

20tab

20tab

@20tab

paulox.net

paolo@melchiorre.org

pauloxnet

paolomelchiorre

@pauloxnet