

Contributing to Django

How I learned to stop worrying and just try to
fix an ORM Bug

First DjangoCon?

Introduction

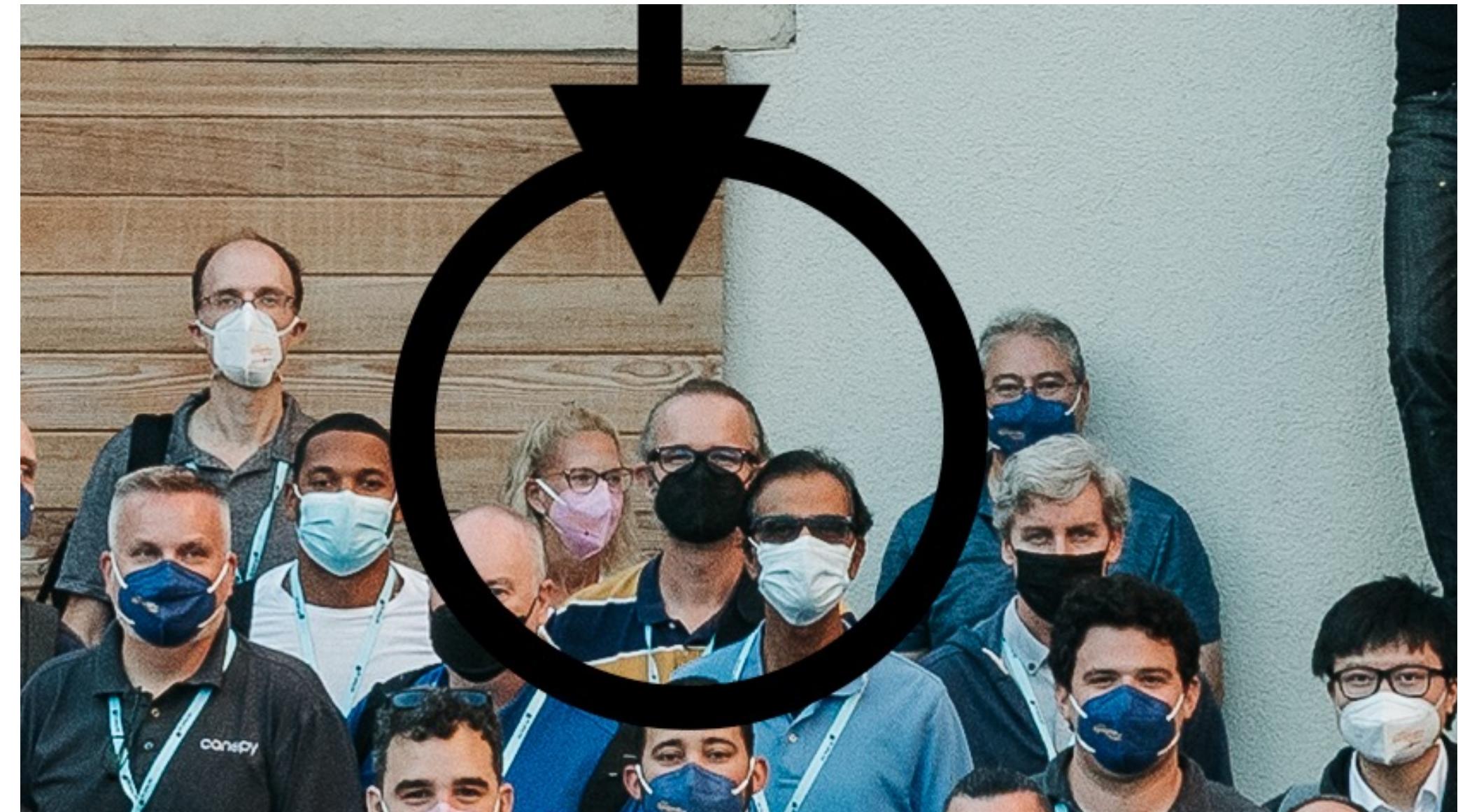
Ryan Cheley



Trying to get to DjangoCon US

- 2018 
- 2019 
- 2020 
- 2021 
- 2022  









Criteria

- Old?
 - January of 2009
- Straightforward(ish)
 - SQLite
- Ticket 10070
 - Title: Named parameters not working on raw sql queries with sqlite
 - Reported by Matias Surdi

Owner: changed from *nobody* to *Ryan Cheley*
Status: new -> Assigned

I'm at DjangoCon US and I'm looking at this Ticket



My Trac Comment

- raw_query\tests
- tests that already exist
- available
- **appear** to be testing
- the ticket is asking for



Comment

- Thanks for looking into this. I can get back to work now :)
 - By Matias Surdi
 - The Reporter of the Issue!

Letting everyone know!



The B is Silent @ryancheley · Oct 20, 2022

I closed a ticket while at the [#DjangoConUS2022](#) sprints with Simon Charette! <https://t.co/UTR1sOnE3A>

0

18

↑

ID 1583206004744867841

[View on Twitter](#)







But then ...

Another Comment

- Broken
- mostly for Oracle and other backends

Another Comment

- Sqlite backend
- supports_paramstyle_pyformat
- borks

What is supports_paramstyle_pyformat?

- Flag
- support 'pyformat' style

What is supports_paramstyle_pyformat?

- ("... %(name)s ...", {'name': value})
- SQLite this was set to False







Feelings ...







Community

Your Web Framework Needs You!

- Ticket difficulty
 - Time
 - Thought
 - Love

Your Web Framework Needs You!

- The Review Process can be challenging
 - Same Process
 - Not Personal
 - Make the code better

Your Web Framework Needs You!

- You can do it!
- You are qualified!

The World Expert

- Wondering
- being worked
- initial time-boxed investigation
- you'll be the world expert ™

Working on the Ticket
... again

Write down what you learned 

Replicate the Bug 

Read some docs 

Write some code 

Test the Code 



Settings.py with psql connection string

```
 DATABASES = {  
     "default": {  
         "ENGINE": "django.db.backends.postgresql",  
         ...  
     }  
 }
```

Steps to Reproduce the Bug: Postgres

```
>>> from django.db import connection  
>>> c = connection.cursor()  
>>> c.execute("select app_label from django_content_type where id =  
1")  
>>> c.execute("select app_label from django_content_type where id =  
%(id)s", {'id':'1'})
```

Steps to Reproduce the Bug: Postgres

- Runs without error

Settings.py with sqlite3 connection string

```
DATABASES = {  
    "default": {  
        "ENGINE": "django.db.backends.sqlite3",  
        ...  
    }  
}
```

Steps to Reproduce the Bug: SQLite

```
>>> from django.db import connection  
>>> c = connection.cursor()  
>>> c.execute("select app_label from django_content_type where id =  
1")  
>>> c.execute("select app_label from django_content_type where id =  
%(id)s", {'id':'1'})
```

```
Traceback (most recent call last):
  File "/Users/ryan/PycharmProjects/tatisjr/venv/lib/python3.9/site-packages/django/db/backends/utils.py", line 89, in _execute
    return self.cursor.execute(sql, params)
  File "/Users/ryan/PycharmProjects/tatisjr/venv/lib/python3.9/site-packages/django/db/backends/sqlite3/base.py", line 301, in execute
    return Database.Cursor.execute(self, query, params)
sqlite3.OperationalError: near "%": syntax error
```

The above exception was the direct cause of the following exception:

```
Traceback (most recent call last):
  File "<console>", line 1, in <module>
  File "/Users/ryan/PycharmProjects/tatisjr/venv/lib/python3.9/site-packages/django/db/backends/utils.py", line 89, in _execute
    return super().execute(sql, params)
  File "/Users/ryan/PycharmProjects/tatisjr/venv/lib/python3.9/site-packages/django/db/backends/utils.py", line 127, in execute
    return self._execute_with_wrappers(
  File "/Users/ryan/PycharmProjects/tatisjr/venv/lib/python3.9/site-packages/django/db/backends/utils.py", line 148, in _execute
    return executor(sql, params, many, context)
  File "/Users/ryan/PycharmProjects/tatisjr/venv/lib/python3.9/site-packages/django/db/backends/utils.py", line 76, in execute
    return self.cursor.execute(sql, params)
  File "/Users/ryan/PycharmProjects/tatisjr/venv/lib/python3.9/site-packages/django/db/utils.py", line 91, in __exit__
    raise dj_exc_value.with_traceback(traceback) from exc_value
  File "/Users/ryan/PycharmProjects/tatisjr/venv/lib/python3.9/site-packages/django/db/backends/utils.py", line 89, in _execute
    return self.cursor.execute(sql, params)
  File "/Users/ryan/PycharmProjects/tatisjr/venv/lib/python3.9/site-packages/django/db/backends/sqlite3/base.py", line 301, in execute
    return Database.Cursor.execute(self, query, params)
django.db.utils.OperationalError: near "%": syntax error
```

Workaround

- SQLite supports named parameters
- Different Syntax

```
>>> c.execute("select name from inventory_host where id=:id", {'id': '1'})  
>>> Host.objects.raw("select * from inventory_host where id=:id", {'id': '1'})
```

- Consistent

Workaround

```
>>> from django.db import connection  
>>> c = connection.cursor()  
>>> c.execute("select app_label from django_content_type where id =  
1")  
>>> c.execute("select * from django_content_type where id =  
:id", {'id': '1'})
```

Replicate the Bug

Write down what you learn

Read some docs

Write some code

Test the Code



Results of Research

- Stack Trace Error Message

```
File "/Users/Ryan/PycharmProjects/tatisjr/venv/lib/python3.9/site-
```

```
packages/django/db/backends/sqlite3/
base.py", line 357, in execute
```

```
    return Database.Cursor.execute(self, query, params)
```

Results of Research

- SQLiteCursorWrapper:
 - execute
 - executemany
 - convert_query

Results of Research

```
def execute(...):
    ...
    query = self.convert_query(query, names=param_names)
    ...

def executemany(...):
    ...
    query = self.convert_query(query, names=param_names)
    ...
```

Results of Research

```
def convert_query(self, query):
    return FORMAT_QMARK_REGEX.sub("?", query).replace("%%", "%")
```

Results of Research

- BUT
- `FORMAT_QMARK_REGEX = _lazy_re_compile(r"(?<!%)%s")`

Results of Research

- `_lazy_re_compile(regex, flags=0)`
- Two parameters
 - regex
 - flags
 - re.I
 - re.S

Results of Research: What have we learned?

- execute() method
- convert_query() method
- executemany() method

✓ Replicate the Bug 🐞

✓ Read some docs 📖

Write down what you learn

Write some code ✎

Test the Code



Ideas for a fix

- `select * from django_content_type where id = %(id)s", {'id': '1'}`
 - fails
- `select * from django_content_type where id = :id", {'id': '1'}`
 - succeeds

Ideas for a fix

- Regular Expression

Ideas for a fix

```
select * from django_content_type where id = %(id)s", {'id': '1'}
```

```
select * from django_content_type where id = :id", {'id': '1'}
```

My Regular Expression

- `select * from django_content_type where id = %(id)s", {'id': '1'}`
- `select * from django_content_type where id = %(id", {'id': '1'})`
- `select * from django_content_type where id = :id", {'id': '1'}`

My Regular Expression

- `query = re.sub("\)s", "", re.sub("%\(", ":", query))`

My Regular Expression

- posted idea

Feedback from Shia Berger

This looks quite fragile

What if string includes `)`s`?

Try naming-dict

```
naming_dict = { param: f":{param}" for param in param_names}  
query = query % naming_dict
```

Feedback from Shia Berger

- Take a look at the Oracle backend
- but at the time I knew what I was doing there

Feedback from Simon Charette

- Avoid using Regex
- Try implementing `__getitem__`
- Ensures `_missing_param_` message
- instead of `KeyError` message

Original convert_query

```
def convert_query(self, query):
    return FORMAT_QMARK_REGEX.sub("?", query).replace("%%", "%")
```

Incorporating Feedback

- Feedback from Shia

```
naming_dict = { param: f":{param}" for param in param_names}  
query = query % naming_dict
```

Incorporating Feedback

- General Idea

```
naming_dict = { param: f":{param}" for param in param_names}  
query = query % naming_dict
```

```
args = {k: "%s" % k for k in params}  
query %= args
```

My initial Code

```
def execute(self, query, params=None):  
    ...  
    if hasattr(params, "keys"):  
        args = {k: ":%s" % k for k in params}  
        query = query % args  
    query = self.convert_query(query)  
    ...
```

My initial Code

```
def executemany(self, query, param_list):
    ...
    param_list = [p for p in param_list]
    try:
        if hasattr(param_list[0], "keys"):
            args = {k: "%s" % k for k in param_list[0]}
            query = query % args
    except IndexError:
        pass
    query = self.convert_query(query)
    ...
```

 Replicate the Bug 

 Read some docs 

 Write some code 

Write down what you learn

Test the Code 



Tests

- Current tests
- New tests (if needed)

Tests: Make sure current tests pass

- supports_paramstyle_pyformat = False
- RawQueryTests
 - test_pyformat_params
 - test_query_representation

Tests: Make sure current tests pass

```
@skipUnlessDBFeature("supports_paramstyle_pyformat")
test_pyformat_params(self):
```

...

Tests: Make sure current tests pass

- Add my proposed code
- Change `supports_paramstyle_pyformat` to be `True`

Tests: Write New Tests

- Not needed

Tests

```
./runtests.py -k test_pyformat_params -k  
test_query_representation
```

Tests: Make sure current tests pass

- supports_paramstyle_pyformat = False
- test_pyformat_params
- test_query_representation

Tests: Make sure current tests pass

```
Testing against Django installed in '/Users/ryan/github/django/django' with up to 8 processes
```

```
Found 2 test(s).
```

```
Creating test database for alias 'default'...
```

```
System check identified no issues (17 silenced).
```

```
S.
```

```
Ran 2 tests in 0.028s
```

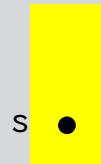
Tests: Make sure current tests pass

```
Testing against Django installed in '/Users/ryan/github/django/django' with up to 8 processes
```

```
Found 2 test(s).
```

```
Creating test database for alias 'default'...
```

```
System check identified no issues (17 silenced).
```



```
Ran 2 tests in 0.028s
```

Possible States Testing Matrix

	Flag = False	Flag = True
Original Code	1 test should pass 1 test should be skipped	1 test should pass 1 test should fail
Updated Code	1 tests should pass 1 test should be skipped	2 tests should pass 

*Flag = `supports_paramstyle_pyformat`

Testing

- Run the tests to check new behavior
- Run entire test suite

✓ Replicate the Bug 🐞

✓ Read some docs 📖

✓ Write some code ✎

✓ Test the Code 💧

Write down what you learned 📄

The rebels, the troublemakers,
the round pegs in the square holes.
The ones who see things differently.
They're not fond of rules. And they
have no respect for the status quo.
You can quote them, disagree with
them, glorify or vilify them. About
the only thing you can't do is ignore
them. Because they're always changing.

Public Notes

What are Public Notes

- Issue
- Work through
 - problem

Public Notes

- Tips
- Hints
- Breadcrumbs
- What ever you want to call them

Scientific Method



My Public Notes

Chrome File Edit View History Bookmarks Profiles Tab Window Help Sat Aug 12 11:02 AM

Troubleshooting Django Ticket | regex101: build, test, and debu... | CLI text processing with GNU g... | Extract Company Name: Python | Convert curl commands to cod... | Fixed #10070 -- Added suppor... +

github.com/ryancheley/public-notes/issues/1

Open Issues

ryancheley / public-notes

Type to search | [Edit](#) [New issue](#)

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Troubleshooting Django Ticket 10070 #1

[Closed](#) ryancheley opened this issue on Oct 30, 2022 · 39 comments

ryancheley commented on Oct 30, 2022

Django Ticket 10070 reports an issue "Named parameters not working on raw sql queries with sqlite"
This is my attempt to try and figure out what is going on.
Side note: I initially marked this ticket as done at Django Con US 2022 as it seemed to no longer be a bug based on review of test methods, but about a week after it was closed, it was re-opened and shown to be reproducible.

ryancheley commented on Oct 30, 2022

Owner Author

Steps to reproduce:
First, let's look at a non-SQLite database and see what happens

```
>>> from django.db import connection
>>> c = connection.cursor()
>>> c.execute("select app_label from django_content_type where id = 1")
>>> c.execute("select app_label from django_content_type where id = %(id)s", {'id':1'})
```

This runs without error.
Now, let's try with connection to a SQLite database

```
>>> from django.db import connection
>>> c = connection.cursor()
>>> c.execute("select app_label from django_content_type where id = 1")
>>> c.execute("select app_label from django_content_type where id = %(id)s", {'id':1'})
```

Assignees: ryancheley

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Create a branch for this issue or link a pull request.

Notifications: Customize [Unsubscribe](#)
You're receiving notifications because you're watching this repository.

1 participant: ryancheley

[Lock conversation](#) [Pin issue](#)

The fix

Incorporating Feedback

- Hint from Shia

```
naming_dict = { param: f":{param}" for param in param_names}  
query = query % naming_dict
```

Incorporating Feedback

- General Idea

```
naming_dict = { param: f":{param}" for param in param_names}  
query = query % naming_dict
```

```
args = {k: "%s" % k for k in params}  
query %= args
```

My initial Code

```
def execute(self, query, params=None):  
    ...  
    if hasattr(params, "keys"):  
        args = {k: ":%s" % k for k in params}  
        query = query % args  
    query = self.convert_query(query)  
    ...
```

My initial Code

```
def executemany(self, query, param_list):
    ...
    param_list = [p for p in param_list]
    try:
        if hasattr(param_list[0], "keys"):
            args = {k: "%s" % k for k in param_list[0]}
            query = query % args
    except IndexError:
        pass
    query = self.convert_query(query)
    ...
```

Original convert_query

```
def convert_query(self, query):
    return FORMAT_QMARK_REGEX.sub("?", query).replace("%%", "%")
```

Issue Comment by Nick Pope

- Duplication
- Potential for incorrect conversion
 - %(value)s to :value
 - Also convert %%s to %s
 - Incorrect conversion to ‘?’

Issue Comment by Nick Pope

- Potential for poor performance
- Materialized Generator
- Pull Request to my Pull Request

PR from Nick Pope

```
def convert_query(self, query, *, names=None):
    if names is None:
        return FORMAT_QMARK_REGEX.sub("?", query).replace("%%", "%")
    else:
        return query % {name: f":{name}" for name in names}
```

PR from Nick Pope

```
def execute(self, query, params=None):  
    ...  
    names = list(params) if isinstance(params, Mapping) else None  
    query = self.convert_query(query, names=names)  
    ...
```

PR from Nick Pope

```
def executemany(self, query, param_list):
    ...
    peekable, param_list = tee(iter(param_list))
    if (params := next(peekable, None)) and isinstance(params, Mapping):
        names = list(params)
    else:
        names = None

    query = self.convert_query(query, names=names)
    ...
```

Full Diff

```
-     def convert_query(self, query):
391 + def convert_query(self, query, *, param_names=None):
392 +     if param_names is None:
393 +         # Convert from "format" style to "qmark" style.
394 +         return FORMAT_QMARK_REGEX.sub("?", query).replace("%%", "%")
395 +     else:
396 +         # Convert from "pyformat" style to "named" style.
397 +         return query % {name: f":{name}" for name in param_names}
```

Full Diff

```
372     def execute(self, query, params=None):
373         if params is None:
374             return Database.Cursor.execute(self, query)
375 -         query = self.convert_query(query)
375 +     # Extract names if params is a mapping, i.e. "pyformat" style is used.
376 +     param_names = list(params) if isinstance(params, Mapping) else None
377 +     query = self.convert_query(query, param_names=param_names)
378         return Database.Cursor.execute(self, query, params)
379
```

Full Diff

```
def executemany(self, query, param_list):
-     query = self.convert_query(query)
+     # Extract names if params is a mapping, i.e. "pyformat" style is used.
+     # Peek carefully as a generator can be passed instead of a list/tuple.
+     peekable, param_list = tee(iter(param_list))
+     if (params := next(peekable, None)) and isinstance(params, Mapping):
+         param_names = list(params)
+     else:
+         param_names = None
+     query = self.convert_query(query, param_names=param_names)
return Database.Cursor.executemany(self, query, param_list)
```

BONUS!

Python SQLite Docs Update!

`sqlite3.paramstyle`

String constant stating the type of parameter marker formatting expected by the `sqlite3` module. Required by the DB-API. Hard-coded to "qmark".

Note: The `sqlite3` module supports both "qmark" and "numeric" DB-API parameter styles, because that is what the underlying SQLite library supports. However, the DB-API does not allow multiple values for the "paramstyle" attribute

Python SQLite Docs Update!

Note: The `sqlite3` module supports both "qmark", "numeric" and "**named**" DB-API parameter styles, because that is what the underlying SQLite library supports. However, the DB-API does not allow multiple values for the "paramstyle" attribute

Python SQLite Docs Update!

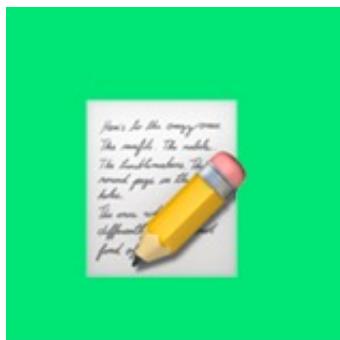
Note: The **named** DB-API parameter style is also supported

The community



Shia Berger

- Identified fragility
- Regex should be avoided
- Starting point



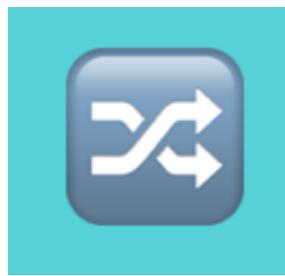
Simon Charette

- Gave great Keynote at DjangoCon US 2022 on the State of the ORM
- Awesome introduction on ORM structure at Sprints
- Update to the docs!



Nick Pope

- PR on my PR to help improve the code
- Remove Code Duplication



Mariusz Felisiak

- Simplified comments
- Merged the PR

Experience

- I learned a TON about
 - SQLite
 - The ORM
 - Python

Experience

- Public Notes
 - Upgrading OS on Linux
 - Python 3.11 on Raspberry Pi
 - SSH Keys

Contributions Since

- Django Packages
 - Documentation improvements
 - Code Reviews

Contributions Since

- Implemented Django at my employer
 - Admin
 - MS SQL

Lessons

The ORM can seem **BIG** and
SCARY

The Code for Django can seem **BIG** and
SCARY

But remember

The Django ORM ...

Is Python

In fact, all of Django ...

Is Python

Looking at Tickets

- Look At 
- Read 
- Write 

Is Python

[GitHub Login](#) [DjangoProject Login](#) [Preferences](#) [API](#)

[View Tickets](#)

[Reports](#)

[Timeline](#)

[Wiki](#)

[Search](#)

Custom Query (1012 matches)



django

The web framework for
perfectionists with deadlines.

Documentation

Triaging tickets









So remember ...

Your Framework needs YOU

Community

Sprints 2023

- Development Sprints
- Contribution Sprints

Acknowledgements

- Katie McLaughlin
- Web Developer Team
 - Bookie
 - Chris
 - Jason
 - Jon
- Abigail Cheley

Thank you

Find me on ...

Platform	QR Code
Mastodon https://mastodon.social/@ryancheley	
GitHub https://github.com/ryancheley/	
LinkedIn https://www.linkedin.com/in/ryan-cheley/	

Reference Links

Item
Your Web Framework Needs You! (Slide 33) https://www.youtube.com/watch?v=8eYM1uPKg7c
Increase your productivity on personal projects with comprehensive docs and automated tests – DCUS (Slide 88) https://www.youtube.com/watch?v=GLkRK2rJGB0
My Public Notes (Slide 93) https://github.com/ryancheley/public-notes/issues/1
Keynote: State of the Object-Relational Mapping (ORM) with Simon Charette (Slide 116) https://www.youtube.com/watch?v=HNIGFrIBI8o
Django Triaging Tickets (Slide 137) https://docs.djangoproject.com/en/dev/internals/contributing/triaging-tickets/