

# A Related Matter:

Optimizing your webapp by using django-debug-toolbar,  
`select_related()`, and `prefetch_related()`

**Christopher Adams**

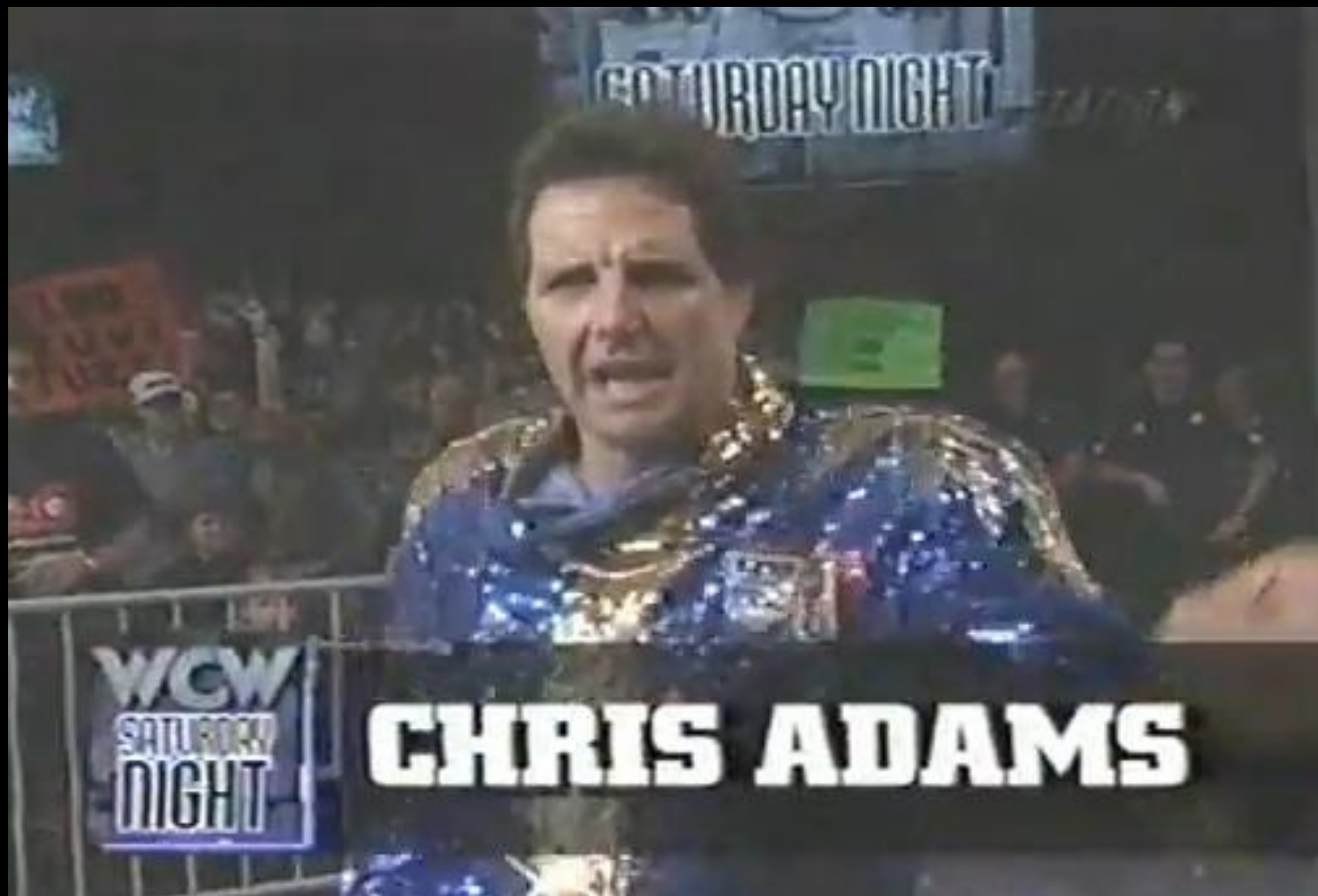
DjangoCon 2024

[github.com/adamsc64/a-related-matter](https://github.com/adamsc64/a-related-matter)

[christopheradams.info](http://christopheradams.info)

# Christopher Adams

- Currently at GitHub, previously at Venmo
- @adamsc64
- I'm not Chris Adams (@acdha), who works at Library of Congress
- Neither of us are "The Gentleman" Chris Adams (90's-era Professional Wrestler)



The Django logo, featuring the word "django" in a white, lowercase, sans-serif font, centered within a dark green rounded rectangle.

django

Django is great

But Django is really a set of tools



# Tools are great

But tools can be used in good or bad ways

# The Django ORM:

## A set of tools

# Manage your own expectations for tools

- Many people approach a new tool with broad set of expectations as to what they think it will do for them.
- This may have little correlation with what the project actually has implemented.

As amazing as it  
would be if they did...



# Unicorns don't exist



# The Django ORM: An abstraction layer

# Abstraction layers

- Great because they take us away from the messy details
- Risky because they take us away from the messy details



# Don't forget

You're far from the ground

# The QuerySet API

QuerySets are Lazy

QuerySets are  
Immutable

Lazy: Does not evaluate  
until it needs to



Immutable: Never  
itself changes

# Each a new QuerySet, none hit the database

- `queryset = Model.objects.all()`
- `queryset = queryset.filter(...)`
- `queryset = queryset.values(...)`

# Hits the database (QuerySet is “evaluated”):

- `queryset = list(queryset)`
- `queryset = queryset[:]`
- `for model_object in queryset:`
- `if queryset:`

Our app:  
blogs hosting site

# Models

```
class Blog(models.Model):  
    submitter = models.ForeignKey('auth.User')  
  
class Post(models.Model):  
    blog = models.ForeignKey('blog.Blog', related_name='posts')  
    likers = models.ManyToManyField('auth.User')  
  
class PostComment(models.Model):  
    submitter = models.ForeignKey('auth.User')  
    post = models.ForeignKey('blog.Post', related_name='comments')
```

# List View

```
def blog_list(request):  
    blogs = Blog.objects.all()  
    return render(request, "blog/blog_list.html", {  
        "blogs": blogs,  
    })
```

# List Template

```
1 {% for blog in blogs %}
2   <li>
3     <a href="{% url 'blog_detail' blog.id %}">
4       {{ blog.name }}
5     </a>
6     | submitted by {{ blog.submitter.username }}
7   </li>
8 {% endfor %}
```

# Blogs:

- [dolor](#) | submitted by James
- [Duis](#) | submitted by Thomas
- [qui](#) | submitted by Isabella
- [irure](#) | submitted by Jack
- [magna](#) | submitted by Lachlan
- [occaecat](#) | submitted by Emma
- [reprehenderit](#) | submitted by James
- [Excepteur](#) | submitted by Oliver
- [non](#) | submitted by Noah
- [dolor](#) | submitted by William
- [veniam,](#) | submitted by Oliver
- [proident,](#) | submitted by Ella
- [aute](#) | submitted by Olivia
- [non](#) | submitted by William
- [magna](#) | submitted by Thomas
- [ut](#) | submitted by Lachlan
- [laborum.](#) | submitted by Thomas
- [fugiat](#) | submitted by Olivia
- [Lorem](#) | submitted by Olivia
- [culpa](#) | submitted by Cooper
- [non](#) | submitted by Thomas
- [sint](#) | submitted by Emily
- [adipiscing](#) | submitted by Isabella



# Detail View

```
def blog_detail(request, blog_id):  
    blog = get_object_or_404(Blog, id=blog_id)  
    posts = Post.objects.filter(blog=blog)  
    return render(request, "blog/blog_detail.html", {  
        "blog": blog,  
        "posts": posts,  
    })
```

# Detail Template

```
1 <h1>Blog '{{ blog.name }}' by {{ blog.submitter.username }}:</h1>
2 <ul>
3 {% for post in posts %}
4   <li>Post '{{ post.name }}'
5     {% if post.likers.exists %}
6     (liked by
7     {% for liker in post.likers.all %}
8       {{ liker.username }}
9     {% endfor %})
10    {% endif %}
11    <ul>
12      {% for comment in post.comments.all %}
13        <li>Comment by {{ comment.submitter.username }}:
14          {{ comment.body|truncatechars:60 }}</li>
15      {% endfor %}
16    </ul>
17  </li>
18 {% endfor %}
```

## Blog 'magna' by Thomas:

- Post 'proident,' (liked by Lachlan Cooper William superuser )
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by James: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Oliver: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lachlan: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lucas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Thomas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
- Post 'exercitation' (liked by James )
  - Comment by Emma: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by William: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Thomas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lily: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Ava: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by William: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
- Post 'qui' (liked by Lachlan James Oliver Ella Lily Jack )
  - Comment by Sophia: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Oliver: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lily: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lucas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by James: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Ava: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
- Post 'commodo' (liked by Lucas Isabella Ava Emma )

What SQL queries are  
happening when I do  
{action}?

# Solution 1: django logging

```
$ python manage.py shell
```

```
>>> import logging
>>> l = logging.getLogger(
    'django.db.backends'
)
>>> l.setLevel(logging.DEBUG)
>>> l.addHandler(logging.StreamHandler())
```

```
In [18]: queryset.aggregate(Avg('id'))
(0.005) SELECT AVG("blog_blog"."id") AS "id__avg" FROM "blog_blog"; args=(); alias=default
Out[18]: {'id__avg': 25.5}
```

# Solution 2: db all-statement logging

```
# postgresql.conf  
log_statement = 'all'
```

```
# my.cnf  
general_log = 1
```

# Django Debug Toolbar

- [Installation](#)
  - [Getting the code](#)
  - [Quick setup](#)
  - [Explicit setup](#)
- [Configuration](#)
  - [DEBUG\\_TOOLBAR\\_PATCH\\_SETTINGS](#)
  - [DEBUG\\_TOOLBAR\\_PANELS](#)
  - [DEBUG\\_TOOLBAR\\_CONFIG](#)
- [Tips](#)
  - [The toolbar isn't displayed!](#)
  - [Using the toolbar offline](#)
  - [Performance considerations](#)
- [Panels](#)
  - [Default built-in panels](#)
  - [Non-default built-in panels](#)
  - [Third-party panels](#)
  - [API for third-party panels](#)
- [Commands](#)
  - `debugsqlshell`
- [Change log](#)
  - [1.2](#)
  - [1.1](#)
  - [1.0](#)
- [Contributing](#)

# Installation

- `pip install django-debug-toolbar==4.4.6`
- Conditional Installation
- So, in `settings.py`:

```
if DEBUG:  
    MIDDLEWARE += ['debug_toolbar.middleware.DebugToolbarMiddleware']
```



First view:  
The blog list page

# Blogs:

- [dolor](#) | submitted by James
- [Duis](#) | submitted by Thomas
- [qui](#) | submitted by Isabella
- [irure](#) | submitted by Jack
- [magna](#) | submitted by Lachlan
- [occaecat](#) | submitted by Emma
- [reprehenderit](#) | submitted by James
- [Excepteur](#) | submitted by Oliver
- [non](#) | submitted by Noah
- [dolor](#) | submitted by William
- [veniam,](#) | submitted by Oliver
- [proident,](#) | submitted by Ella
- [aute](#) | submitted by Olivia
- [non](#) | submitted by William
- [magna](#) | submitted by Thomas
- [ut](#) | submitted by Lachlan
- [laborum.](#) | submitted by Thomas
- [fugiat](#) | submitted by Olivia
- [Lorem](#) | submitted by Olivia
- [culpa](#) | submitted by Cooper
- [non](#) | submitted by Thomas
- [sint](#) | submitted by Emily
- [adipiscing](#) | submitted by Isabella

## Time

CPU: 561.78ms (588.91ms)

## Settings

## Headers

## Request

BLOG\_LIST

## SQL

52 QUERIES IN 41.55ms

## Static files

0 FILES USED

## Templates

BLOG/BLOG\_LIST.HTML

## Cache

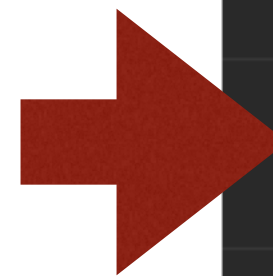
0 CALLS IN 0.00ms

## Signals

8 RECEIVERS OF 12 SIGNALS

# Blogs:

- [dolor](#) | submitted by James
- [Duis](#) | submitted by Thomas
- [qui](#) | submitted by Isabella
- [irure](#) | submitted by Jack
- [magna](#) | submitted by Lachlan
- [occaecat](#) | submitted by Emma
- [reprehenderit](#) | submitted by James
- [Excepteur](#) | submitted by Oliver
- [non](#) | submitted by Noah
- [dolor](#) | submitted by William
- [veniam,](#) | submitted by Oliver
- [proident,](#) | submitted by Ella
- [aute](#) | submitted by Olivia
- [non](#) | submitted by William
- [magna](#) | submitted by Thomas
- [ut](#) | submitted by Lachlan
- [laborum.](#) | submitted by Thomas
- [fugiat](#) | submitted by Olivia
- [Lorem](#) | submitted by Olivia
- [culpa](#) | submitted by Cooper
- [non](#) | submitted by Thomas
- [sint](#) | submitted by Emily
- [adipiscing](#) | submitted by Isabella



## Time

CPU: 561.78ms (588.91ms)

## Settings

## Headers

## Request

BLOG\_LIST

## SQL

52 QUERIES IN 41.55ms

## Static files

0 FILES USED

## Templates
































BLOG/BLOG\_LIST.HTML

## Cache

0 CALLS IN 0.00ms

## Signals

8 RECEIVERS OF 12 SIGNALS

Query	Timeline
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 1	
 SELECT ... FROM "blog_blog"	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 17	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 15	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 11	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 2	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 19	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 4	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 17	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 14	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 12	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 3	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 14	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 10	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 5	
 SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 3	

# The N+1 Query Problem

- An N+1 query problem occurs when a system runs one query to fetch a list of items (the "1"), and then runs an additional query (the "N") for each item in that list to fetch related data.
- This leads to inefficient performance, as it can result in a large number of queries being executed unnecessarily (each query has latency cost).
- It is unfortunately an easy bug to introduce using ORM frameworks like Django or Rails.





```
SELECT "auth_user"."id", "auth_user"."password",  
"auth_user"."last_login", "auth_user"."is_superuser",  
"auth_user"."username", "auth_user"."first_name",  
"auth_user"."last_name", "auth_user"."email", "auth_user"."is_staff",  
"auth_user"."is_active", "auth_user"."date_joined" FROM "auth_user"  
WHERE "auth_user"."id" = 4
```

0.67

Sel

Expl

**Connection:** default

```
/Users/chris/coding/djangocon2014_project/adamsc64_djangocon2014/blog/views.py in blog_list(10)  
    "blogs": blogs,
```

```
7         <a href="{% url 'blog_detail' blog.id %}">  
8             {{ blog.name }}  
9         </a>  
10         | submitted by {{ blog.submitter.username }}  
11     </li>  
12 {% endfor %}  
13 </ul>
```

```
/Users/chris/coding/djangocon2014_project/adamsc64_djangocon2014/blog/templates/blog/blog_list.html
```



```
SELECT "auth_user"."id", "auth_user"."password",  
"auth_user"."last_login", "auth_user"."is_superuser",  
"auth_user"."username", "auth_user"."first_name",  
"auth_user"."last_name", "auth_user"."email", "auth_user"."is_staff",  
"auth_user"."is_active", "auth_user"."date_joined" FROM "auth_user"  
WHERE "auth_user"."id" = 4
```

0.67

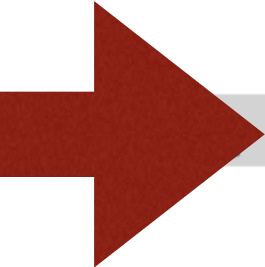
Sel

Expl

Connection: default

```
/Users/chris/coding/djangocon2014_project/adamsc64_djangocon2014/blog/views.py in blog_list(10)  
    "blogs": blogs,
```

```
7         <a href="{% url 'blog_detail' blog.id %}">  
8             {{ blog.name }}  
9         </a>  
10         | submitter.username }}  
11     </li>  
12 {% endfor %}  
13 </ul>
```



```
/Users/chris/coding/djangocon2014_project/adamsc64_djangocon2014/blog/templates/blog/blog_list.html
```

# List Template

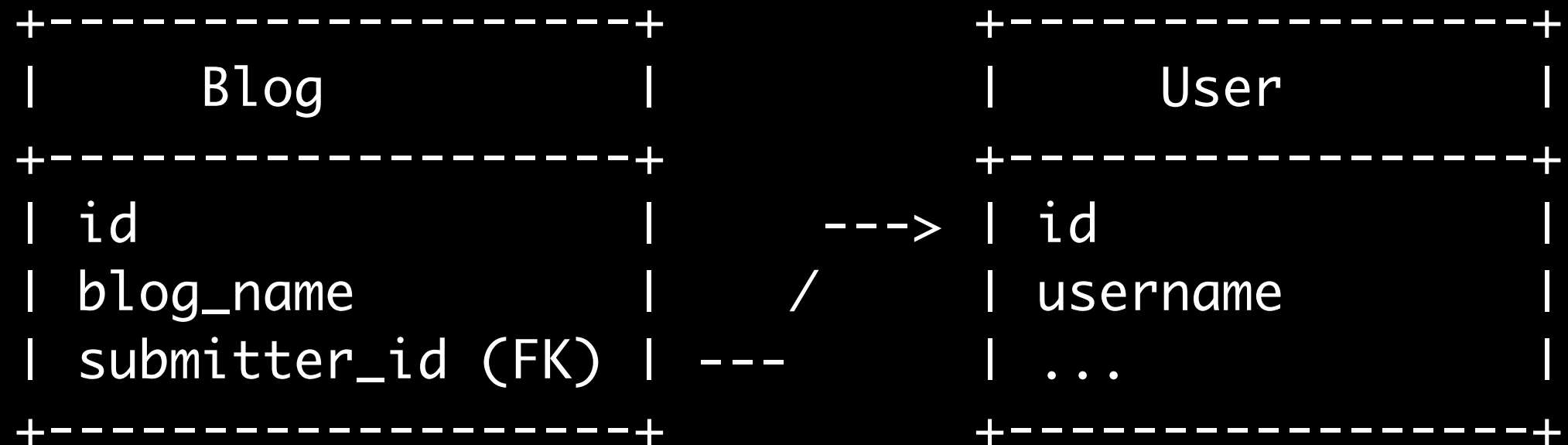
```
1 {% for blog in blogs %}
2   <li>
3     <a href="{% url 'blog_detail' blog.id %}">
4       {{ blog.name }}
5     </a>
6     | submitted by {{ blog.submitter.username }}
7   </li>
8 {% endfor %}
```



# select\_related()

- **select\_related** uses SQL joins to include fields from related objects in a single SELECT statement.
- This allows Django to fetch related objects in the **same database query**, improving efficiency.
- However, select\_related is only effective for single-valued relationships, such as **foreign key** and **one-to-one** relationships.

# ForeignKey



Multiple blogs can be associated with one user.

# List View

```
def blog_list(request):  
    blogs = Blog.objects.all()  
    blogs = blogs.select_related("submitter")  
    return render(request, "blog/blog_list.html", {  
        "blogs": blogs,  
    })
```

# SQL queries from 1 connection



default

5.49 ms (2 queries)

Query	Timeline	Time (ms)	Action
<div><div></div><div><div></div><div><div></div><div>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 1</div></div></div></div>		3.08	<div>Sel</div> <div>Expl</div>
<div><div></div><div><div></div><div><div></div><div>SELECT ... FROM "blog_blog" INNER JOIN "auth_user" ON ( "blog_blog"."submitter_id" = "auth_user"."id" )</div></div></div></div>		2.40	<div>Sel</div> <div>Expl</div>

Hide »

Versions

DJANGO 1.6.7

Time

CPU: 44.35ms (53.91ms)

Settings

Headers

Request

BLOG\_LIST

SQL

2 QUERIES IN 5.49MS

Static files

0 FILES USED

Templates



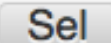
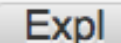



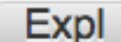
BLOG/BLOG\_LIST.HTML

# SQL queries from 1 connection



default

5.49 ms (2 queries)

Query	Timeline	Time (ms)	Action
 <b>SELECT ... FROM</b> "auth_user" <b>WHERE</b> "auth_user"."id" = 1		3.08	 
 <b>SELECT ... FROM</b> "blog_blog" <b>INNER JOIN</b> "auth_user" <b>ON</b> ( "blog_blog"."submitter_id" = "auth_user"."id" )		2.40	 

Hide »

Versions

DJANGO 1.6.7

Time

CPU: 44.35ms (53.91ms)

Settings

Headers

Request

BLOG\_LIST

SQL

2 QUERIES IN 5.49MS

Static files

0 FILES USED

Templates

BLOG/BLOG\_LIST.HTML

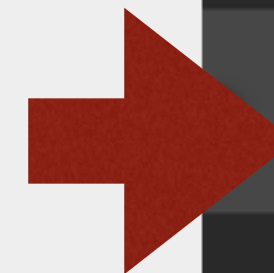
# SQL queries from 1 connection



default

5.49 ms (2 queries)

Query	Timeline	Time (ms)	Action
<b>SELECT ... FROM</b> "auth_user" <b>WHERE</b> "auth_user"."id" = 1		3.08	
<b>SELECT ... FROM</b> "blog_blog" <b>INNER JOIN</b> "auth_user" <b>ON</b> ( "blog_blog"."submitter_id" = "auth_user"."id" )		2.40	



Hide »

Versions

DJANGO 1.6.7

Time

CPU: 44.35ms (53.91ms)

Settings

Headers

Request

BLOG\_LIST

SQL

2 QUERIES IN 5.49MS

Static files

0 FILES USED

Templates

BLOG/BLOG\_LIST.HTML

Second view:  
The blog detail page

# Blog 'magna' by Thomas:

- Post 'proident,' (liked by Lachlan Cooper William superuser )
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by James: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Oliver: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lachlan: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lucas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Thomas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
- Post 'exercitation' (liked by James )
  - Comment by Emma: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by William: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Thomas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lily: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Ava: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by William: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
- Post 'qui' (liked by Lachlan James Oliver Ella Lily Jack )
  - Comment by Sophia: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Oliver: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lily: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lucas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by James: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Ava: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
- Post 'commodo' (liked by Lucas Isabella Ava Emma )

Hide »

## Versions

DJANGO 1.6.7

## Time

CPU: 464.95ms (500.62ms)

## Settings

## Headers

## Request

BLOG\_DETAIL

## SQL

44 QUERIES IN 42.43MS

## Static files

0 FILES USED

## Templates

BLOG/BLOG\_DETAIL.HTML

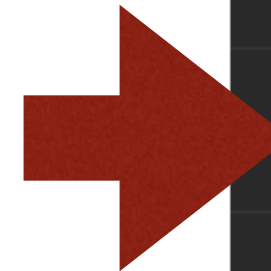
## Cache

0 CALLS IN 0.00MS



## Blog 'magna' by Thomas:

- Post 'proident,' (liked by Lachlan Cooper William superuser )
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by James: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Oliver: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lachlan: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lucas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Thomas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
- Post 'exercitation' (liked by James )
  - Comment by Emma: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by William: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Thomas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lily: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Ava: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by William: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
- Post 'qui' (liked by Lachlan James Oliver Ella Lily Jack )
  - Comment by Sophia: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Oliver: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by superuser: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lily: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Lucas: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by James: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
  - Comment by Ava: Lorem ipsum dolor sit amet, consectetur adipiscing elit, ...
- Post 'commodo' (liked by Lucas Isabella Ava Emma )



Hide »

### Versions

DJANGO 1.6.7

### Time

CPU: 464.95ms (500.62ms)

### Settings

### Headers

### Request

BLOG\_DETAIL

### SQL

44 QUERIES IN 42.43MS

### Static files

0 FILES USED

### Templates

BLOG/BLOG\_DETAIL.HTML

### Cache

0 CALLS IN 0.00MS

# SQL queries from 1 connection



default

42.56 ms (44 queries)

Query	Timeline	Time (ms)	Action
SELECT ... FROM "blog_blog" WHERE "blog_blog"."id" = 15		2.45	
SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 1		2.45	
SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 15		0.76	
SELECT ... FROM "blog_post" WHERE "blog_post"."blog_id" = 15		1.50	
SELECT ... FROM "auth_user" INNER JOIN "blog_post_likers" ON ( "auth_user"."id" = "blog_post_likers"."user_id" ) WHERE "blog_post_likers"."post_id" = 60 LIMIT 1		1.75	
SELECT ... FROM "auth_user" INNER JOIN "blog_post_likers" ON ( "auth_user"."id" = "blog_post_likers"."user_id" ) WHERE "blog_post_likers"."post_id" = 60		1.57	
SELECT ... FROM "blog_postcomment" WHERE "blog_postcomment"."post_id" = 60		1.26	
SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 1		0.77	
SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 17		0.95	
SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 14		0.73	
SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 19		0.84	
SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 1		0.99	
SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 18		0.67	
SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 15		1.18	
SELECT ... FROM "auth_user" INNER JOIN "blog_post_likers" ON ( "auth_user"."id" = "blog_post_likers"."user_id" ) WHERE "blog_post_likers"."post_id" = 59 LIMIT 1		1.06	
SELECT ... FROM "auth_user" INNER JOIN "blog_post_likers" ON ( "auth_user"."id" = "blog_post_likers"."user_id" ) WHERE "blog_post_likers"."post_id" = 59		1.27	
SELECT ... FROM "blog_postcomment" WHERE "blog_postcomment"."post_id" = 59		0.64	



# SQL queries from 1 connection



default

42.56 ms (44 queries)

Query	Timeline	Time (ms)	Action
<b>SELECT ... FROM "blog_blog" WHERE "blog_blog"."id" = 15</b>		2.45	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 1</b>		2.45	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 15</b>		0.76	
<b>SELECT ... FROM "blog_post" WHERE "blog_post"."blog_id" = 15</b>		1.50	
<b>SELECT ... FROM "auth_user" INNER JOIN "blog_post_likers" ON ( "auth_user"."id" = "blog_post_likers"."user_id" ) WHERE "blog_post_likers"."post_id" = 60 LIMIT 1</b>		1.75	
<b>SELECT ... FROM "auth_user" INNER JOIN "blog_post_likers" ON ( "auth_user"."id" = "blog_post_likers"."user_id" ) WHERE "blog_post_likers"."post_id" = 60</b>		1.57	
<b>SELECT ... FROM "blog_postcomment" WHERE "blog_postcomment"."post_id" = 60</b>		1.26	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 1</b>		0.77	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 17</b>		0.95	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 14</b>		0.73	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 19</b>		0.84	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 1</b>		0.99	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 18</b>		0.67	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 15</b>		1.18	
<b>SELECT ... FROM "auth_user" INNER JOIN "blog_post_likers" ON ( "auth_user"."id" = "blog_post_likers"."user_id" ) WHERE "blog_post_likers"."post_id" = 59 LIMIT 1</b>		1.06	
<b>SELECT ... FROM "auth_user" INNER JOIN "blog_post_likers" ON ( "auth_user"."id" = "blog_post_likers"."user_id" ) WHERE "blog_post_likers"."post_id" = 59</b>		1.27	
<b>SELECT ... FROM "blog_postcomment" WHERE "blog_postcomment"."post_id" = 59</b>		0.64	

 **SELECT** "auth\_user"."id", "auth\_user"."password",  
"auth\_user"."last\_login", "auth\_user"."is\_superuser",  
"auth\_user"."username", "auth\_user"."first\_name",  
"auth\_user"."last\_name", "auth\_user"."email", "auth\_user"."is\_staff",  
"auth\_user"."is\_active", "auth\_user"."date\_joined" **FROM** "auth\_user"  
**INNER JOIN** "blog\_post\_likers" **ON** ( "auth\_user"."id" =  
"blog\_post\_likers"."user\_id" ) **WHERE** "blog\_post\_likers"."post\_id" = 58

1.06

Sel

Expl

**Connection:** default

/Users/chris/coding/djangocon2014\_project/adamsc64\_djangocon2014/blog/views.py in **blog\_detail(21)**  
"posts": posts,

```
6         <li>Post '{{ post.name }}'  
7             {% if post.likers.exists %}  
8             (liked by  
9             {% for liker in post.likers.all %}  
10                 '{{ liker.username }}'  
11             {% endfor %})  
12             {% endif %}
```

/Users/chris/coding/djangocon2014\_project/adamsc64\_djangocon2014/blog/templates/blog/blog\_detail.html

**SELECT** "auth\_user"."id", "auth\_user"."password",  
"auth\_user"."last\_login", "auth\_user"."is\_superuser",  
"auth\_user"."username", "auth\_user"."first\_name",  
"auth\_user"."last\_name", "auth\_user"."email", "auth\_user"."is\_staff",  
"auth\_user"."is\_active", "auth\_user"."date\_joined" **FROM** "auth\_user"  
**INNER JOIN** "blog\_post\_likers" **ON** ( "auth\_user"."id" =  
"blog\_post\_likers"."user\_id" ) **WHERE** "blog\_post\_likers"."post\_id" = 58

1.06

Sel

Expl

Connection: default

/Users/chris/coding/djangocon2014\_project/adamsc64\_djangocon2014/blog/views.py in **blog\_detail(21)**  
"posts": posts,

```
6         <li>Post '{{ post.name }}'  
7             {% if post.likers.exists %}  
8             (liked by  
9             {% for liker in post.likers.all %}  
10                '{{ liker.username }}'  
11             {% endfor %})  
12             {% endif %}
```

/Users/chris/coding/djangocon2014\_project/adamsc64\_djangocon2014/blog/templates/blog/blog\_detail.html

# prefetch\_related()

- **prefetch\_related** is useful to efficiently query on many-to-many or "reverse" foreign key relationships
- Without this function, Django does a query for each user who likes a comment, which causes an **N+1 problem**.
- Using **prefetch\_related**, Django fetches the posts, then the users who like them in **only two queries**. It then "links" them in Python.

# Detail View

```
def blog_detail(request, blog_id):  
    blog = get_object_or_404(Blog, id=blog_id)  
    posts = Post.objects.filter(blog=blog)  
    posts = posts.prefetch_related(  
        "likers"  
    )  
    return render(request, "blog/blog_detail.html", {  
        "blog": blog,  
        "posts": posts,  
    })
```

**SELECT** "auth\_user"."id", "auth\_user"."password",  
"auth\_user"."last\_login", "auth\_user"."is\_superuser",  
"auth\_user"."username", "auth\_user"."first\_name",  
"auth\_user"."last\_name", "auth\_user"."email", "auth\_user"."is\_staff",  
"auth\_user"."is\_active", "auth\_user"."date\_joined" **FROM** "auth\_user"  
**WHERE** "auth\_user"."id" = 14

0.73

Sel

Expl

**Connection:** default

/Users/chris/coding/djangocon2014\_project/adamsc64\_djangocon2014/blog/views.py in **blog\_detail(21)**  
"posts": posts,

```
12         {% endif %}
13     <ul>
14         {% for comment in post.comments.all %}
15             <li>Comment by {{ comment.submitter.username }}:
16                 {{ comment.body|truncatechars:60 }}</li>
17         {% endfor %}
18     </ul>
```

/Users/chris/coding/djangocon2014\_project/adamsc64\_djangocon2014/blog/templates/blog/blog\_detail.html



**SELECT** "auth\_user"."id", "auth\_user"."password",  
"auth\_user"."last\_login", "auth\_user"."is\_superuser",  
"auth\_user"."username", "auth\_user"."first\_name",  
"auth\_user"."last\_name", "auth\_user"."email", "auth\_user"."is\_staff",  
"auth\_user"."is\_active", "auth\_user"."date\_joined" **FROM** "auth\_user"  
**WHERE** "auth\_user"."id" = 14

0.73


Sel

Expl

**Connection:** default

/Users/chris/coding/djangocon2014\_project/adamsc64\_djangocon2014/blog/views.py in **blog\_detail(21)**  
"posts": posts,

```
12         {% endif %}
13     <ul>
14         {% for comment in post.comments.all %}
15         <li>Comment by {{ comment.submitter.username }}:
16             {{ comment.body|truncatechars:60 }}</li>
17         {% endfor %}
18     </ul>
```



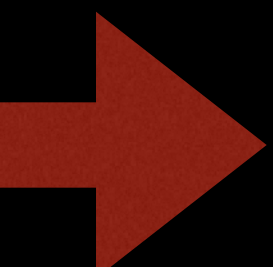
/Users/chris/coding/djangocon2014\_project/adamsc64\_djangocon2014/blog/templates/blog/blog\_detail.html

# Models

```
class Blog(models.Model):  
    submitter = models.ForeignKey('auth.User')  
  
class Post(models.Model):  
    blog = models.ForeignKey('blog.Blog', related_name='posts')  
    likers = models.ManyToManyField('auth.User')  
  
class PostComment(models.Model):  
    submitter = models.ForeignKey('auth.User')  
    post = models.ForeignKey('blog.Post', related_name='comments')
```

# Models

```
class Blog(models.Model):  
    submitter = models.ForeignKey('auth.User')  
  
class Post(models.Model):  
    blog = models.ForeignKey('blog.Blog', related_name='posts')  
    likers = models.ManyToManyField('auth.User')  
  
class PostComment(models.Model):  
    submitter = models.ForeignKey('auth.User')  
    post = models.ForeignKey('blog.Post', related_name='comments')
```



# Detail View

```
def blog_detail(request, blog_id):  
    blog = get_object_or_404(Blog, id=blog_id)  
    posts = Post.objects.filter(blog=blog)  
    posts = posts.prefetch_related(  
        "comments__submitter", "likers",  
    )  
    return render(request, "blog/blog_detail.html", {  
        "blog": blog,  
        "posts": posts,  
    })
```

# "comments\_\_submitter"

- "comments": i.e., all the comments for each post.
  - A reverse-relation: the related\_name="comments" in the PostComment model
- For each comment, it also fetches the user ("\_\_submitter") who made that comment.
- This prefetch instruction reduces queries and makes the retrieval of related data more efficient.

## SQL queries from 1 connection



default

12.16 ms (7 queries)

Query	Timeline	Time (ms)	Action
<b>SELECT ... FROM "blog_blog" WHERE "blog_blog"."id" = 15</b>		3.58	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 1</b>		2.09	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" = 15</b>		0.69	
<b>SELECT ... FROM "blog_post" WHERE "blog_post"."blog_id" = 15</b>		1.06	
<b>SELECT ... FROM "blog_postcomment" WHERE "blog_postcomment"."post_id" IN (60, 59, 58, 57)</b>		1.70	
<b>SELECT ... FROM "auth_user" WHERE "auth_user"."id" IN (1, 2, 3, 4, 7, 8, 9, 10, 14, 15, 16, 17, 18, 19)</b>		0.95	
<b>SELECT ... FROM "auth_user" INNER JOIN "blog_post_likers" ON ( "auth_user"."id" = "blog_post_likers"."user_id" ) WHERE "blog_post_likers"."post_id" IN (60, 59, 58, 57)</b>		2.09	

Hide »

Versions



DJANGO 1.6.7

Time



CPU: 86.12ms (100.47ms)

Settings



Headers



Request



BLOG\_DETAIL

SQL



7 QUERIES IN 12.16MS

Static files



0 FILES USED

Templates



BLOG/BLOG\_DETAIL.HTML

Cache



# Summary

- The QuerySet API methods `select_related()` and `prefetch_related()` implement best practices to reduce unnecessary queries.
- Use **`select_related()`** for one-to-many or one-to-one relations.
- Use **`prefetch_related()`** for many-to-many or reverse foreign key relations.

# Thanks!

Christopher Adams (@adamsc64)

[github.com/adamsc64/a-related-matter](https://github.com/adamsc64/a-related-matter)

[christopheradams.info](http://christopheradams.info)