A Related Matter:

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

Christopher Adams

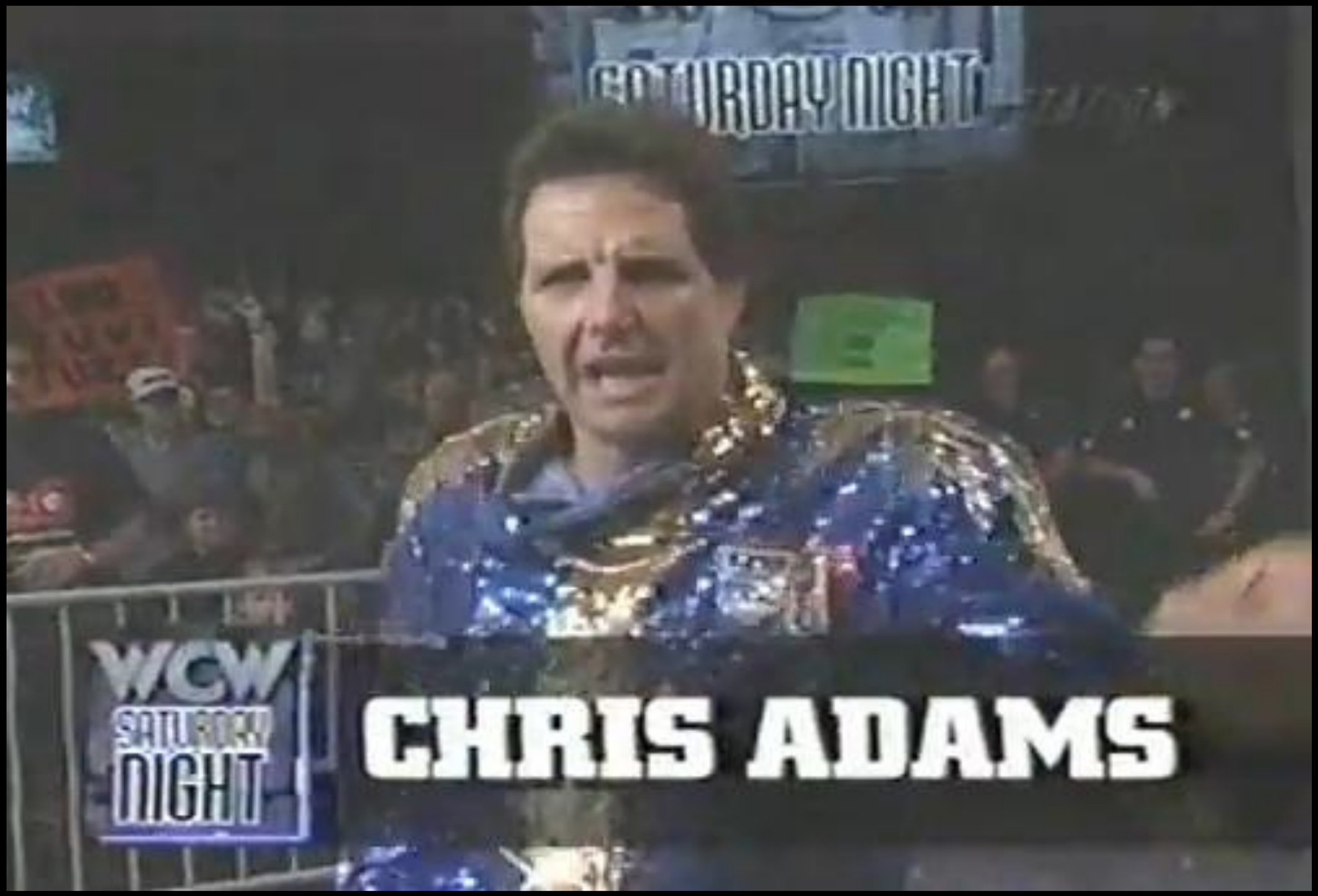
DjangoCon 2024

`github.com/adamsc64/a-related-matter`

`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)


SATURDAY NIGHT
WCW
SATURDAY
NIGHT
CHRIS ADAMS

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 the 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

- 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

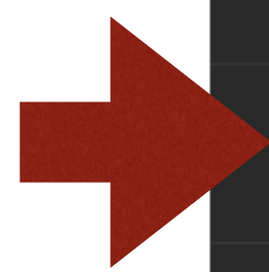
Templates
BLOG/BLOG_LIST.HTML

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.
- 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           | submitt                blog.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

```
+--------------------+          +------------------+
|      Blog          |          |      User        |
+--------------------+          +------------------+
| id                 |     ---> | id               |
| blog_name          |    /     | username         |
| submitter_id (FK)  | ---      | ...              |
+--------------------+          +------------------+
```

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
+ **SELECT ··· FROM** "auth_user" **WHERE** "auth_user"."id" = 1		3.08	Sel Expl
+ **SELECT ··· FROM** "blog_blog" **INNER JOIN** "auth_user" **ON** ("blog_blog"."submitter_id" = "auth_user"."id")		2.40	Sel Expl

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
+ **SELECT ··· FROM** "auth_user" **WHERE** "auth_user"."id" = 1		3.08	Sel Expl
+ **SELECT ··· FROM** "blog_blog" **INNER JOIN** "auth_user" **ON** ("blog_blog"."submitter_id" = "auth_user"."id")		2.40	Sel Expl

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
+ **SELECT ··· FROM** "auth_user" **WHERE** "auth_user"."id" = 1		3.08	Sel Expl
+ **SELECT ··· FROM** "blog_blog" **INNER JOIN** "auth_user" **ON** ("blog_blog"."submitter_id" = "auth_user"."id")		2.40	Sel Expl

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

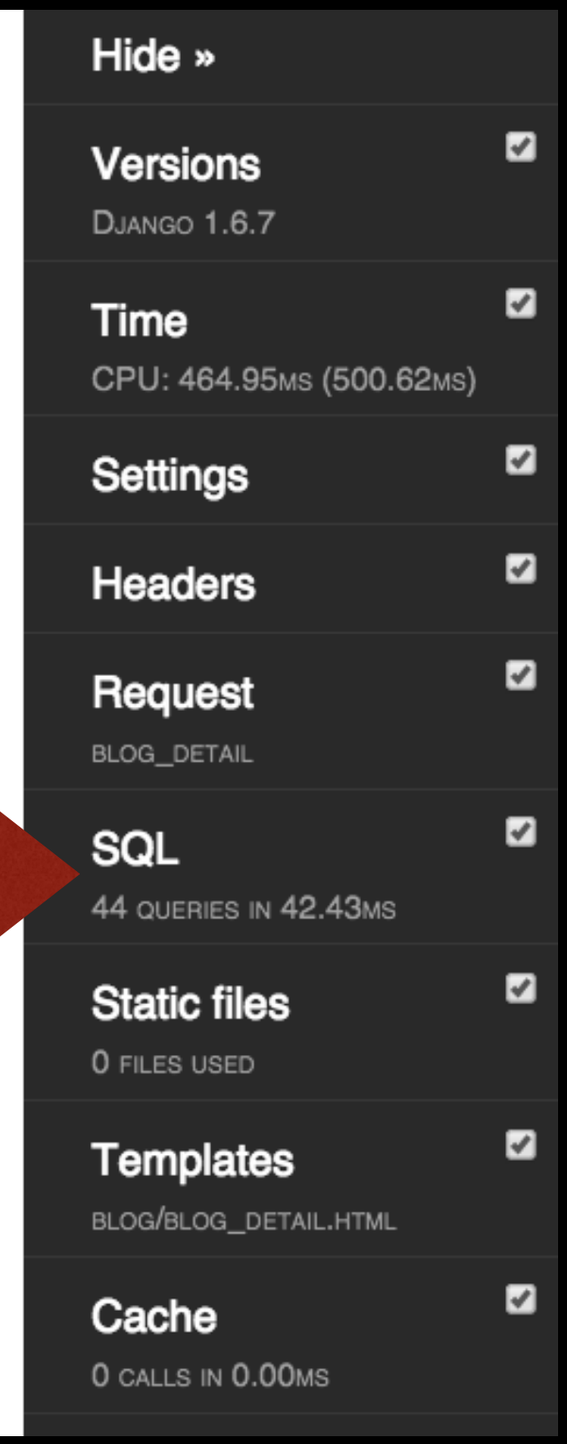
Second view:
The blog detail page

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)

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)



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	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 1		2.45	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 15		0.76	Sel Expl
SELECT ··· **FROM** "blog_post" **WHERE** "blog_post"."blog_id" = 15		1.50	Sel Expl
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	Sel Expl
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	Sel Expl
SELECT ··· **FROM** "blog_postcomment" **WHERE** "blog_postcomment"."post_id" = 60		1.26	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 1		0.77	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 17		0.95	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 14		0.73	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 19		0.84	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 1		0.99	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 18		0.67	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 15		1.18	Sel Expl
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	Sel Expl
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	Sel Expl
SELECT ··· **FROM** "blog_postcomment" **WHERE** "blog_postcomment"."post_id" = 59		0.64	Sel Expl

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	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 1		2.45	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 15		0.76	Sel Expl
SELECT ··· **FROM** "blog_post" **WHERE** "blog_post"."blog_id" = 15		1.50	Sel Expl
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	Sel Expl
SELECT ··· **FROM** "auth_user" **INNER JOIN** "blog_post_likers" **ON** ("auth_user"."id" = "blog_post_likers"."use **ERE** "blog_post_likers"."post_id" = 60		1.57	Sel Expl
SELECT ··· **FROM** "blog_po "blog_postcomment"."post		1.26	Sel Expl
SELECT ··· **FROM** "auth_ = 1		0.77	Sel Expl
SELECT ··· **FROM** "auth_ = 17		0.95	Sel Expl
SELECT ··· **FROM** "auth_u d" = 14		0.73	Sel Expl
SELECT ··· **FROM** "auth_use "id" = 19		0.84	Sel Expl
SELECT ··· **FROM** "auth_user" **WH** user"."id" = 1		0.99	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 18		0.67	Sel Expl
SELECT ··· **FROM** "auth_user" **WHERE** "auth_user"."id" = 15		1.18	Sel Expl
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	Sel Expl
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	Sel Expl
SELECT ··· **FROM** "blog_postcomment" **WHERE** "blog_postcomment"."post_id" = 59		0.64	Sel Expl

- **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             {% fo                post.likers.all %}
10                {{ li                .username }}
11            {% endfor %})
12            {% endif %}
```

/Users/chris/coding/djangocon2014_project/adamsc64_djangocon2014/blog/templates/blog/blog_detail.html

prefetch_related()

- **prefetch_related** is especially useful when dealing with 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 all the users for all comments in a single query and then "links" them in Python.
- This way, instead of running a new query for each comment, it runs just two queries: one for the comment and one for the related users, avoiding an N+1 problem.

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  post.comments.all %}
15              <li>Com  comment.submitter.username }}:
16                  {{ comm .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')
```

"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.

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,
    })
```

SQL queries from 1 connection

default

12.16 ms (7 queries)

Query	Timeline	Time (ms)	Action
SELECT ··· FROM "blog_blog" **WHERE** "blog_blog"."id" = 15		3.58	Sel Expl
SELECT ··· FROM "auth_user" **WHERE** "auth_user"."id" = 1		2.09	Sel Expl
SELECT ··· FROM "auth_user" **WHERE** "auth_user"."id" = 15		0.69	Sel Expl
SELECT ··· FROM "blog_post" **WHERE** "blog_post"."blog_id" = 15		1.06	Sel Expl
SELECT ··· FROM "blog_postcomment" **WHERE** "blog_postcomment"."post_id" **IN** (60, 59, 58, 57)		1.70	Sel Expl
SELECT ··· FROM "auth_user" **WHERE** "auth_user"."id" **IN** (1, 2, 3, 4, 7, 8, 9, 10, 14, 15, 16, 17, 18, 19)		0.95	Sel Expl
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)		2.09	Sel Expl

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

Besides SQL Queries...

- **Cache**: Shows Django cache hits, misses, and the time taken for cache operations.
- **Signals**: Django signals sent during the request-response cycle.
- **Headers and Request Info**: HTTP headers, session data, environment information.
- **Static Files**: information about static media used to render the page

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

christopheradams.info