

DJANGO ORM

BEYOND THE BASICS

Bill Blanchard, Plumb Development LLC

April 28, 2016

AGENDA

- ▶ What are QuerySets? How do they work?
- ▶ QuerySet API methods you might not know about
- ▶ Lookup Types
- ▶ Q objects
- ▶ Further Reading Topics

WHAT IS A QUERYSET?

- ▶ An object that represents a group of SQL queries
- ▶ You can use the QuerySet API to query the database on your application's behalf without knowing raw SQL
- ▶ QuerySets are lazy
- ▶ Raw SQL vs. QuerySets

```
In [1]: users = User.objects.all()
```

```
In [2]: print(users.query)
```

```
SELECT "users_user"."password", "users_user"."last_login",  
"users_user"."is_superuser", "users_user"."username",  
"users_user"."first_name", "users_user"."last_name", "users_user"."email",  
"users_user"."is_staff", "users_user"."is_active", FROM "users_user"
```

QUERYSET API METHODS

- ▶ `get` vs. `filter` vs. `exclude`
- ▶ `order_by`
- ▶ `aggregate`
- ▶ `annotate`
- ▶ `values/values_list`
- ▶ `earliest/latest`

QUERYSET API METHODS

- ▶ Get returns one and only one object

```
>>> Book.objects.get(id=1)
```

```
<Book: 1>
```

- ▶ So use distinct fields

```
>>> Author.objects.get(first_name="Bill")
```

```
MultipleObjectsReturned: get() returned more than one Author --  
it returned more than 20!
```

- ▶ Shortcuts are useful

```
>>> book_object = get_object_or_404(Book, id=1)
```

```
<Book: 1>
```

QUERYSET API METHODS

▶ Filter returns a QuerySet

```
>>> Book.objects.filter(category="Sci-Fi")
```

```
[<Book: 1>, <Book: 2>, <Book: 3>, '...(remaining elements truncated)...']
```

▶ Same as exclude

```
>>> Book.objects.exclude(average_review_stars=1)
```

```
[<Book: 1>, <Book: 2>, <Book: 3>, '...(remaining elements truncated)...']
```

▶ Which can be chained

```
>>> Book.objects.filter(category="Sci-Fi").exclude(average_review_stars=1)
```

```
[<Book: 1>, <Book: 2>]
```

QUERYSET API METHODS

- ▶ Aggregate allows you a quick way to find Min, Max, Avg, Sum, StdDev, Variance

```
>>> Book.objects.aggregate(average_price=Avg('price'))
```

```
{'average_price': 34.35}
```

```
>>> Book.objects.aggregate(highest_stock=Max('quantity_in_stock'))
```

```
{'highest_stock': 350}
```

- ▶ Count is quick and easy

```
>>> Book.objects.filter(category="Sci-Fi").count()
```

QUERYSET API METHODS

- ▶ Annotate allows you to “attach” information to your objects in a QuerySet

```
>>> pubs = Publisher.objects.annotate(num_books=Count('book'))
```

```
>>> pubs
```

```
[<Publisher BaloneyPress>, <Publisher SalamiPress>, ...]
```

```
>>> pubs[0].num_books
```


QUERYSET API METHODS

- ▶ `values` allows you to extract individual attribute data without having to get the whole object

```
>>> Blog.objects.values()
```

```
[{'id': 1, 'name': 'Beatles Blog', 'tagline': 'All the latest Beatles news.'}],
```

```
>>> Blog.objects.values('id', 'name')
```

```
[{'id': 1, 'name': 'Beatles Blog'}]
```

- ▶ `values_list` behaves the same, except it gives tuples instead of dicts

```
>>> Entry.objects.values_list('id', 'headline')
```

```
[(1, 'First entry'), ...]
```

```
>>> Entry.objects.values_list('id', flat=True).order_by('id')
```

```
[1, 2, 3, ...]
```

QUERYSET API METHODS

- ▶ `earliest/latest()` are easy, useful date/datetime methods

```
>>> Entry.objects.latest('pub_date')
```

```
>>> Entry.objects.earliest('pub_date')
```

LOOKUP TYPES

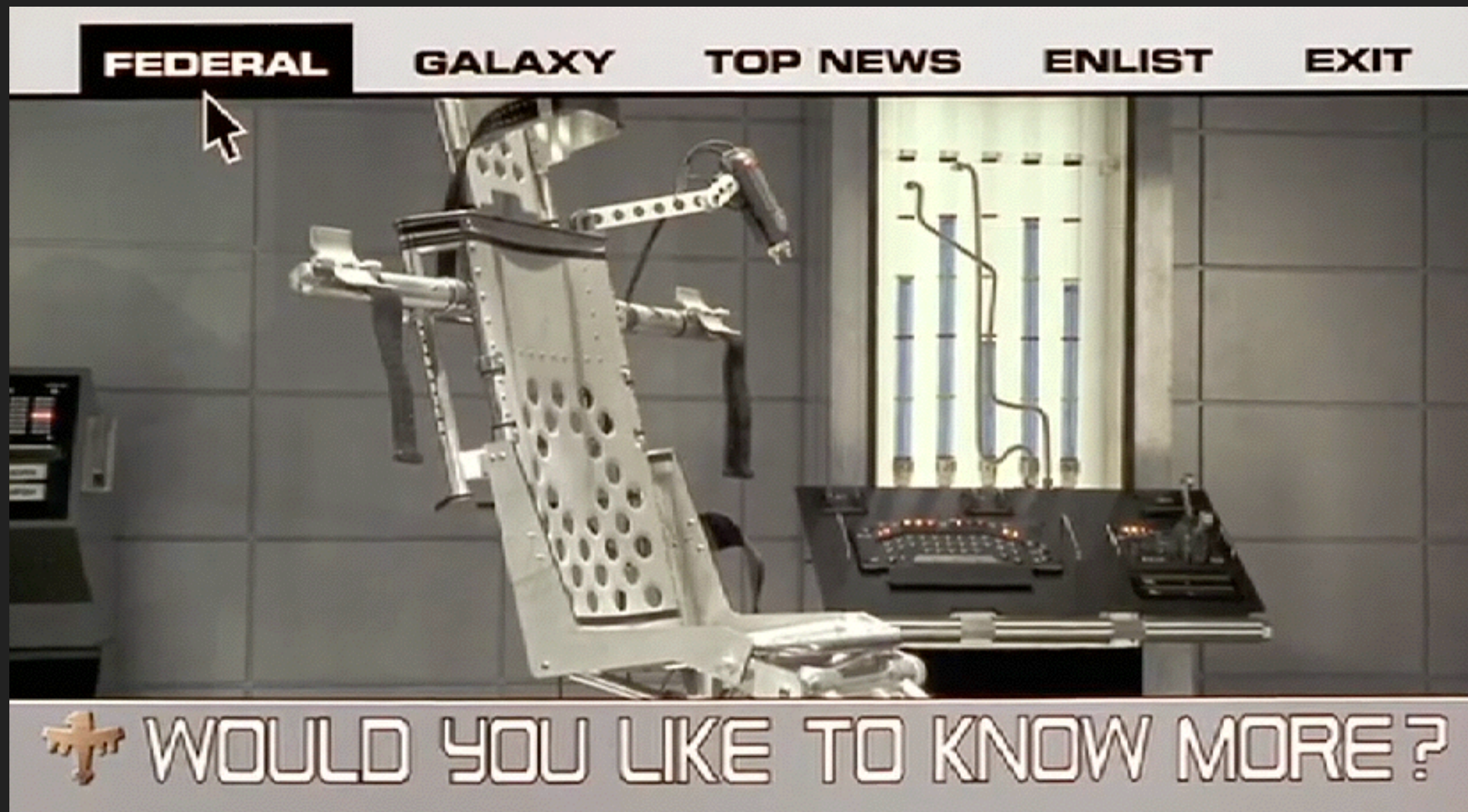
- ▶ (i)exact
- ▶ (i)contains
- ▶ in
- ▶ (i)startswith/(i)endswith
- ▶ lt(e)/gt(e)/range
- ▶ datetime lookup types
- ▶ (i)regex

Q OBJECTS

- ▶ “A Q Object is an object used to encapsulate a collection of keyword arguments”
- ▶ i.e. filter with conditionals (ANDs, ORs)

```
Poll.objects.filter(  
    Q(question__startswith='Who'),  
    Q(pub_date=date(2005, 5, 2)) | Q(pub_date=date(2005, 5, 6))  
)
```

DJANGO ORM: BEYOND THE BASICS



- ▶ QuerySet API docs (methods, lookup types, etc.)
- ▶ Model Meta documentation (built-in and custom methods)
- ▶ QuerySet managers (advanced)
- ▶ F() Objects

QUESTIONS?

THANKS!



BILL BLANCHARD
BILL@PLUMBDEV.COM
TWITTER: @BILLISPLUMB
GITHUB: @CHROMAKEY

* The majority of code samples were taken from the Django docs. We thank the DSF and all the OSS contributors for their hard work and dedication.