Accelerating Information Technology Innovation

http://aiti.mit.edu

Nigeria Summer 2012
Lecture DJ02 – Django Database Intro

MISTI

Database Interaction

Managers

- `Manager` is a class
- It's the interface between the database and django
- Various methods, including `filter()`, `exclude()`, and `order_by()`
- Also has `get_query_set`, which returns a `QuerySet` object

QuerySets

- `QuerySet` is a class
- Does not initiate the database interaction until told to
- Also has similar methods including `filter()`, `exclude()`, and `order_by()`

Getting all data

- `Blog.objects.get_query_set.all()`
- Shorthand: `Blog.objects.all()`
- Gets all the data associated with the model but does NOT execute the query

Filtering Data

- exact: gets an exact match
  - `Blog.objects.filter(title__exact='cool')`
  - `Blog.objects.filter(title='cool')` # __exact is implied
- contains: find if a match is contained inside a field
  - `Blog.objects.filter(blog_text__contains='cool')`
- icontains: case insensitive contains
  - `Blog.objects.filter(author__icontains='smith')`
- More here: https://docs.djangoproject.com/en/1.3/ref/models/querysets/#field-lookups

Ordering

- `Blog.objects.order_by('-pub_date', 'title')`
  - First orders by `pub_date` in descending order (hence the negative sign). If there are `pub_dates` that are equivalent, then `title` is ordered in ascending order.

Values

- `Blog.objects.values()`
  - Returns a `ValueQuerySet`, which returns a list of dictionaries *when executed*
- `Blog.objects.values('title', 'body')`
  - Returns only the fields `title` and `body` in the dictionary

Distinct

- `Blog.objects.distinct()`
  - If there are any duplicate rows, only one is returned
  - This will rarely work like this, because you often will already have a distinct field, like an id
- `Blog.objects.values('title', 'body').distinct()`
  - This will get all unique `title-body` combinations
  - Notice the **chaining** here

Slicing

- `Blog.objects.all()[:5]`
  - Gets the first 5 blog objects
  - The limit happens in the sql query
    - ex: SELECT * FROM users LIMIT 5

Get

- Gets a single row
- raises `MultipleObjectsReturned` if more than one object was found. The `MultipleObjectsReturned` exception is an attribute of the `model` class.
- raises a `DoesNotExist` exception if an object wasn't found for the given parameters. This exception is also an attribute of the `model` class.

`Get` continued

- `Blog.objects.get(id=5)`
  - Returns a single `QuerySet` if there is a row that exists, otherwise an error ensues
- `Blog.objects.filter(id=5)[0]`
  - Similar, except no exceptions are thrown

When are QuerySets Evaluated?

- Iteration

```
for e in Entry.objects.all():
    print e.headline
```

- Boolean

```
if Entry.objects.filter(headline="Test"):
   print "There is at least one Entry with the
headline Test"
```

Lookups that span relationships

- `Blog.objects.filter(comment__title__contains='Lennon')`
  - Retrieves all `Blog` objects with a comment whose `title` contains `'Lennon'`

Other Syntax

URLs

```
urlpatterns = patterns('',
    url(r'^$', 'blog.views.home'),
    url(r'^list/(₩d+)?$', 'blog.views.blog_list'),
    url(r'^search/(.*)$', 'blog.views.blog_search'),
    url(r'^(detail|info)/(?P<id>₩d+)/((?P<showComments>.*)/) ?$',
'blog.views.blog_detail'),
)
```

Views

```
def store_list(request, limit=100):
    store_list = Store.objects.all()[:limit]
    print store_list # [<Store: phones>, <Store: food>]
    return HttpResponse('going to give a list')
```