Accelerating Information Technology Innovation

http://aiti.mit.edu

Ghana Summer 2012
Lecture DJ04 – Django Views
Simple Diagram

Browser

urls.py

Model/Database

View

HTTP Request

HTTP Request

Data Request

Data

HTTP Response

HTTP Response
Web Browsing

Users submit requests to websites through:
  – Desktop web browsers and applications
  – Smartphone web browsers and applications

• Django calls a view function associated with the URL

• User defines a map between the URL and the view function
Hello World! Django

Sample URLConf and view function below

```python
from django.conf.urls.defaults import patterns, include, url
urlpatterns = patterns('',
    url(r'^$', 'views.hello_world')
)
```

```python
from django.http import HttpResponse
def hello_world(request):
    return HttpResponse("Hello world!")
```
Hello World! Django

Let’s see the results!

- `python manage.py runserver`

- Cool! How do we make more exciting websites?
  - Models- Store useful information
  - Templates- Produce dynamic pages
  - Views- We can do a lot more than “Hello world!”
Checkpoint

• Which file should we modify (and how) if we want to see “Hello World!” at the following URL?
  – `http://127.0.0.1:8000/hello_world`

• Which file(s) should we modify (and how) if we want to see “Hello Mars” at the following URL?
  – `http://127.0.0.1:8000/hello_mars`
Views

• How can we customize the view?
  – parameters from the URL (regexps)
Views

• Parameters in the URL
  – Regular Expressions specify the rules for URL’s
  • Resources available online to learn more

• Consider a universal greeting:

```python
def urlpatterns = patterns('',
    url(r'hello_(\w+)/$', 'views.hello_anyone'),
)```
Views

• Universal greeting

urlpatterns = patterns('',
    url(r'^hello\w+/$', 'views.hello_anyone'),
)

• Our view function has two parameters
  • request (HttpRequest object)
  • planet (string object)

    from django.http import HttpResponse
    def hello_anyone(request, planet):
        my_response = "Hello " + str(planet)
        return HttpResponse(my_response)
Views

- Testing it out...

Hello jupiter!

Hello pluto!
Database Interaction

• We want to be able to display information from our database tables as well!
Getting all data

Blog.objects.all()

Gets all the data associated with the model but does NOT execute the query

It’s not a list, it’s an instance of QuerySet
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

```python
# This list contains a Blog object.
>>> Blog.objects.filter(name__startswith='Beatles')
[<Blog: Beatles Blog>]

# This list contains a dictionary.
>>> Blog.objects.filter(name__startswith='Beatles').values()
[{'id': 1, 'name': 'Beatles Blog', 'tagline': 'All the latest Beatles news.'}]
```
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.distinct('title', 'body')`
  - This will get all unique title–body combinations
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.

• Raises a `DoesNotExist` exception if an object wasn't found for the given parameters.
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**
  ```python
  for e in Entry.objects.all():
      print e.headline
  ```

- **Boolean**
  ```python
  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'
Views Example

def get_titles(request, limit=100):
    book_list = Book.objects.all()[:limit]
    response = 'List of titles is:'
    for b in book_list:
        response+=str(b.title)
    return HttpResponse(response)