MIT AITI
Django Lab 5: Views and urls

In this lab we will learn about views and urls.py. We will create another page for our blog app that displays blogs containing a given search term. We will define which URLs route to this page, write the view to get the data, and write the template to display the page. Read through the whole lab before starting.

If you get stuck, take a look at these resources:

1. Lecture slides
2. Previous labs
3. Other group members
4. Django documentation
    a. Making Queries: https://docs.djangoproject.com/en/1.5/topics/db/queries/
    b. QuerySets: https://docs.djangoproject.com/en/dev/ref/models/querysets/
    c. Views: https://docs.djangoproject.com/en/1.5/topics/http/views/
    d. Urls: https://docs.djangoproject.com/en/1.5/topics/http/urls/
5. Google
6. Instructors

Steps:

1. Open up your `blog` app
   `$ cd ~/Desktop/myblog/blog`

2. Open urls.py. It should contain this text. It might not contain the bolded line – be sure to type it in! If it contains other lines too, that's ok – just leave them in and ignore them!

```
from django.conf.urls import patterns, include, url
urlpatterns = patterns('',
    url(r'^$', 'blog.views.home'),
    url(r'^list/(\d+)?$', 'blog.views.blog_list'),
url(r'^(detail|info)/(?P<id>\d+)/((?P<showComments>.*)/)?$'
, 'blog.views.blog_detail'),

#url(r'^search/(happy)$', 'blog.views.blog_search'),
)
```

3. Uncomment the line
`url(r'^search/(happy)$', 'blog.views.blog_search'),`

This line says that whenever someone goes to the url localhost:8000/blog/search/happy, it will use the view "blog_search". **This view does not exist yet – you will write it later in the lab!**

4. We want people to be able to search for any term, not just "happy". Change the word happy to instead be a regular expression that matches **any text (any character, any number of times).**

5. Now open the `views.py`. At the end, add the code:

```
def blog_search(request, term):
  blog_list =
```

6. Now we will describe how to write this view function. See the other views functions for examples of how to do these steps.
First, finish the line "`blog_list=`" by selecting all blogs that contain the search term (given by the variable `term`) in their body. The matching should be case insensitive. See the views reference page for how to do this.

7. Add a line of code to the function to select the template "blog/search.html". We will make this template later in the lab.

8. Add a line that names the variables needed in the template. The template will need the search term in a variable called "term" and the blog list in a variable called "blog_list".

9. Finally, add a line that returns the template and context. See the other views functions to see how to do this.

10. Now create the template that will display this page. Call the template search.html. Make sure to put it in the correct directory!
Here's the code for the template. It assumes that you have 2 variables passed in from the view: one called "term" that is the search term, and one called "blog_list" that is a list for matching blogs.

```
{% extends 'blog/base.html' %}
{% block content %}

<h2>Search term: {{ term }}</h2>
<h2>Matching blog posts:</h2>

{% for blog in blog_list %}
<h3><a href="/blog/detail/{{blog.id}}">{{blog.title}}</a></h3>
<div>{{ blog.body|linebreaks }}</div>
{% endfor %}

{% endblock %}
```

11. Run your server (python manage.py runserver), and go to localhost:8000/blog/search/<your term here>
Replace <your term here> with a word that appears in some of your blogs. See if the correct blogs are displayed!

Views - QuerySets

To do this…	**Use this code**
Basics	
Get all instances of model	Blog.objects.all()
Gets the first 5 blog objects	Blog.objects.all()[:5]
Get one object by id	Blog.objects.get(id=4)
Filters: Used to get a subset of data that matches some condition	
Exact match	Blog.objects.filter(title='cool')
Contains (find if a match is contained in a field)	Blog.objects.filter(body__contains='cool') # remember to use **2 underscores**!
Contains (case insensitive)	Blog.objects.filter(author__icontains='smith') # remember to use **2 underscores**!
Ordering results	
Order by a given field: ascending	Blog.objects.order_by('pub_date')
Order by a given field: descending	Blog.objects.order_by('-pub_date')

Regular Expressions

Matching		
Expression	**What it matches**	**Examples**
.	anything	. matches “a”
\d	numbers	\d matches “4”
^	beginning of line	^abc matches “abcd”. ^abc does not match “zabc”
$	end of line	a$ matches “ha”. a$ does not match “hah”
Repeats		
Character	**Matches something occuring…**	**Examples**
*	any number of times	a* matches “”, “a”, and “aaaa”
+	one or more times	a+ matches “a” and “aaaa”
?	zero or one time	a? matches “” and “a”
Symbol	**Means…**	**Example**
()	Grouping	(hello)+ matches “hellohello”
|	Or	(hello|bye) matches “hello” and “bye”