MIT AITI
Mobile, Python, Software Development

Django Blog Project
Part 3: URLs and Views

You will now be adding views to your Blog app!

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/1.4/topics/db/queries/
    b. QuerySets: https://docs.djangoproject.com/en/dev/ref/models/querysets/
    c. Views: https://docs.djangoproject.com/en/1.4/topics/http/views/
    d. Urls: https://docs.djangoproject.com/en/1.4/topics/http/urls/
5. Google
6. Instructors

Steps:

1. Add a url to the `urls.py` file (that is in the same folder as `settings.py`) that will send any url starting with '`blog/`' to a file called urls.py inside of the `blog` app

   Try to figure this out… but answer is at the end of this file

2. Go to your `blog` app (just `cd` to the directory)

3. Create a file called `urls.py`. Copy the code from "instructions/part_3_instructor_code/blog_urls.py" into the file.

4. Now add another `url` that will match a url that follows the pattern:

   "posts/search/" *and then anything*.

   Make sure you capture the *anything*, and pass it to the function `blog.views.post_search`

   For example, `blog/posts/search/gorilla` should capture "gorilla" and call the view function `post_search` with `"gorilla"` as the first parameter.

   Hint: Use regexs! and the other urls should help you figure out what to do.

5. Now edit the `views.py` file. Copy the code from "`instructions/part_3_instructor_code/blog_views.py`" and paste it in to "`blog/views.py`"
6. You need to implement three views: `post_list`, `post_detail`, and `post_search`.
   a) For `post_list`, you should return an HttpResponse containing a textual representation of all of the posts
   b) For `post_detail`, get a single post (identified by the `id` parameter) and return a textual representation of it as an `HttpResponse`. **If** `showComments` is not `False`, get the comments associated with the current blog (the current blog is the blog with the id), and make them part of the `HttpResponse` as well.
   c) For `post_search`, return an `HttpResponse` with a textual representation of all the blogs that contain the search `term`.

By the way, when I say "textual representation", I just mean a string

10. Run your server (`python manage.py runserver`), and go to the admin interface

11. Add some blog posts and comments associated with them

12. Load the following pages, making sure they work:
    a) blog/
    b) blog/posts
    c) blog/posts/1
    d) blog/posts/1/true
    e) blog/posts/search/<your term here>

13. If they look like they are working, but everything is all jammed together and you want it to look more like `terminal` output, read the **Note on HTML** below.

14. If everything looks good, push to heroku.

15. By the way you **should be pushing to github, too, every 2 hours or so**

Answers:

1. Add the line

   ```
   url(r'^blog/', include('blog.urls')),
   ```

   to the list of patterns being assigned to the variable `urlpatterns`

Note on HTML:
The reason that everything is jumbled together is that your browser is trying to *render* the text that you provide as HTML (a language that describes how a webpage is structured).

There are ways around this, here are two easy ones:

First, you can view the **source** of the page that is rendered. Every browser has a way to do this, in firefox you should right-click the page and then click "View Page Source".

Second, and a way that will only work with this django project, is to use a tool that has been provided by the instructors that will adjust your output so that the browser *does not* attempt to render it as HTML, so it will appear exactly as you intend it to. To activate this tool, add the line:

```
'instructors.tools.RawResponseMiddleware',
```

to the tuple called `MIDDLEWARE_CLASSES` in your `settings.py` file.