MIT AITI
Django Lab 7: Users and Registration

In this lab, you will be adding a way to interact with you users in your blog project. The end result will be the following:

- Users can log in and log out
- The blog detail page welcomes the user by name
- Users can add comments if they are logged in (and the author name is automatically filled in)

Part 1: Set up user login system

1. First we'll make an app to handle user registration and logging in/out. Make a new app called `reg`.
   ```
   $ cd ~/Desktop/myblog
   $ django-admin.py startapp reg
   ```
   This will make a new folder called `reg`

2. Edit your `~/Desktop/myblog/myblog/urls.py`. After the line that redirects all `blog/` urls to the `blog/urls.py` file, write a similar line that redirects all `reg/` urls to the `reg/urls.py` file

3. Go into the reg directory
   ```
   $ cd reg
   ```

4. The models file doesn't need to be edited because Django makes a User model by default

5. Create a file in the `reg` directory called `urls.py`. Just like the urls.py in the blog directory is in charge of all urls starting with localhost:8000/blog/, this urls.py file will describe urls starting with localhost:8000/reg/

In reg/urls.py, type:

```
from django.conf.urls import patterns, include, url
urlpatterns = patterns('',
    #your urls here
)
```

6. In `urls.py`, add a line that points `localhost:8000/reg/login/` to the loginView function like so:
   ```
   url(r'^login/$', 'reg.views.loginView'),
   ```

7. Add another line that points logout/ to the `logoutView`

8. Now edit the `views.py` file and put in the following:

```
from django.template import Context, loader
from django.http import HttpResponse, HttpResponseRedirect
from django import forms
from django.contrib.auth import authenticate, login, logout
from django.shortcuts import render_to_response
from django.views.decorators.csrf import csrf_exempt

class LoginForm(forms.Form):
    username = forms.CharField()
    password = forms.CharField(widget=forms.PasswordInput)

@csrf_exempt
def loginView(request):
  if request.method == 'POST':
    #YOUR CODE HERE
  form = LoginForm()
  return render_to_response('reg/login.html', {
    'form': form,
    'logged_in': request.user.is_authenticated()
  })

@csrf_exempt
def logoutView(request):
  logout(request)
  return render_to_response('reg/logout.html')
```

There's a new method we're calling, called `render_to_response`. This is just a shortcut (notice this is from `django.shortcuts` in the import statement), to pass some context to a template. This accomplishes the same thing as the code we were using before (define template, define context, render the template with a context), but is only one line long.

9. Now we will implement code in the `loginView` where it says `#YOUR CODE HERE` This `if` statement will be executed when you press the submit button to log in. Check if the username and password are correct, and if they are, log in the user and refresh the page (i.e. redirect to the current page). You can do this as follows:

- The submitted username is `request.POST['username']`. (Reminder: `request.POST` is a dictionary). You can access the submitted password in the same way, by getting it from the request.POST dictionary.
- You can check if a user put in the correct password using `authenticate(username=`*`my_username`*`, password=`*`my_password`*`)`. (where *my_username* and *my_password* are variables containing the username and password). If this function returns `None`, then authentication failed. Otherwise, it worked.
- You can log the user in by doing `login(request, user)`, where **user is the returned value of the authenticate function**.

10. Now that your view is done, you can edit the templates.
```
$ cd ~/Desktop/myblog/templates
$ mkdir reg
$ cd reg
```

    Make two new template files (the filenames expected are listed in the views). In the file corresponding to the `loginView`, put the following:

```
<a href='/blog/list'>Blog</a>
<form action="." method="post">
{{ form.as_p }}
<input type="submit" value="Submit" />
</form>
```

    This will show the user the login form every time. Instead, change this so that the user is shown "you are already logged in", and a logout link if they are logged in.
    Hint: look at the context you are passing to this view. How can you tell whether a user is logged in?
    Hint: see the next step for an example of how to make a link

11. In the file corresponding to the logoutView, put the following:
```
<a href='/blog/list'>Blog</a><br />
You have been successfully logged out. <br />
<a href='/reg/login'>login</a>
```

12. Test to see if it's working. First, go to your admin page and add a few users. Then, go to localhost:8000/reg/login, and make sure you can log in. Go to localhost:8000/reg/logout and make sure you can log out.

Part 2: Integrate users into your blog:

13. Open up your blog view
```
$ gedit ~/Desktop/myblog/blog/views.py
```

14. In the `blog_detail` method, add a variable called `request` to the Context being passed to the template. This will allow the template to access the view's request object.

15. Now open the template for detail.html. In the template, add a welcome message to the user to display at the top of the page. For example, if I was logged in as "Leah" it could display "Welcome Leah!". To access the user's username, you can use request.user.username. How can you display this value in the template?

16. If you're done with Friday's lab (Forms), do the next step. **Otherwise work on Friday's lab now**.

17. Open up your blog view
```
$ gedit ~/Desktop/myblog/blog/views.py
```

18. Change the CommentForm so it no longer asks for the author's name.

19. In the `blog_detail` method, set a comment's author to be the user's username (As above, you can access this through the `request` variable)