**MIT AITI**
**Python Software Development**


## Lab 07: Users and Registration

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

- Users can log in and log out
- Users can add a comment if they are logged in (and the author is automatically filled in)
- Users can edit only their own comments

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

1. Lecture slides
2. Previous labs
3. Other group members
4. Django documentation
    a. Authentication: https://docs.djangoproject.com/en/dev/topics/auth/
5. Google
6. Instructors

**Part 1: Set up user login system**

1. 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 `urls.py`. After the part that redirects all `blog/` urls to the `blog/urls.py` file, make 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. In `urls.py`, add a line that points all login/ urls to the login_view function like so:
   ```
   url(r'^login/$', 'reg.views.login_view'),
   ```

   Add another line that points logout to the `logout_view`

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

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

Recall the function `render_to_response`, mentioned briefly in Lab 5. 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.

Implement the code in the `login_view` 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).
- The submitted username is `request.POST['username']`. (Reminder: `request.POST` is a dictionary)
- You can check if a user is authenticated using `authenticate(username, password)`. If the return value is `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.

7. Now that your view is done, you can now edit the templates.

```
$ cd ~/Desktop/myblog/templates
$ mkdir reg
```

```
$ cd reg
```

Make two new template files (the filenames are listed in the views). In the file corresponding to the `login_view`, 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.

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

9. 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 and log out.

**Part 2: Integrate users into your blog:**

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

11. You need to make three changes to the `blog_detail` method:
    - change one of the lines such that the `author` field no longer shows up
    - Make sure the `author` is still put into the comment before it is saved. (Hint: the author is the user's username. You need to find out how to get this.)
    - pass the '`request`' parameter to the template by adding it to the context. This will be used in deciding what to show in `detail.html`
    - Change the `comment_edit` method such that an `HttpResponse` with the text "You do not have permission to edit this comment" is returned if the user is not allowed to edit a comment. Only the author is allowed to edit his/her own comment.

12. Finally, edit the `details` template so that only logged-in users can add comments, and users can only edit their own comments.

13. Check that this works properly when you are logged in or logged out.