Django Level Five

Let’s learn something!

PIERIAN DATA

Django Bootcamp

- Welcome to Django Level Five!
- This section of the course will focus on User Authentication.
- So far we've only created applications that assume everyone will see the same page.

Django

- This is one of the aspects of Django where there are lots of built in available tools as well as plenty of external packages that enhance functionality.

Django

- We will focus mainly on the built-in tools:
  - Users and the User Model
  - Permissions
  - Groups
  - Passwords and Authentication
  - Logging In and Out

Django

- Let’s get started!

Passwords

Django Level Five

PIERIAN DATA

Django

- In this lecture we will discuss the general set-up to begin getting ready for User Authentication.
- We'll talk about passwords in general and also discuss some additional library options for security.

Django

- The first thing we need to take care of is setting up our ability to authenticate a User.
- To do this we need to use some built-in apps and make sure they are under the INSTALLED_APPS list in settings.py

Django

- The apps we will use are “django.contrib.auth” and “django.contrib.contenttypes”
- Often these will already be pre-loaded in the list for you.
- Remember to migrate if you added them!

Django

- The next thing we need to do is make sure we store our passwords safely.
- **Never store passwords as plain text!**
- We will begin by using the default PBKDF2 algorithm with an SHA256 hash that is built-in to Django.

Django

- This is quite secure for most applications, it requires a massive amount of computing power to crack it.
- But if you want more security you can upgrade to even more secure hashing algorithms.

Django

- Django makes this really easy, we will also show how to use the bcrypt and Argon2.
- In your virtual environment:
    - **pip install bcrypt**
    - **pip install django[argon2]**
    - Depending on your Django version may already have these installed.

Django

- Inside of settings.py you can then pass in the list of PASSWORD_HASHERS to try in the order you want to try them.
- If for some reason you don't have the library support, eventually you will fall back on to the original PBKDF2

Django

- Sometimes users will also try to use a very weak password, such as “password123”.
- We can also add in validator options to prevent a user from doing that.
- We’ll keep things simple and only require a minimum length for now.

Django

- Let’s now code through the steps of setting up a password system.
- After this we can set-up our User Models and our Registration Forms.

User Models

Django Level Five

PIERIAN DATA

Django

- In this lecture we will discuss how to use Django's built in tools to create User Authorization Models.
- We will also discuss how to set-up media files in your project.

Django

- Previously when we've logged on to the Admin page we've seen that there is already a built-in Authentication and Authorization model set in place.
- In this database there were "Users".
- Let's learn how to use this feature!

Django

- The **User** object has a few key features:
    - Username
    - Email
    - Password
    - First Name
    - Surname

Django

- There are also some other attributes for the **User** object, such as is_active, is_staff, is_superuser.
- Sometimes you will also want to add more attributes to a user, such as their own links or a profile image.

Django

- You can do this in your applications models.py file by creating another class that has a relationship to the **User** class.
- Let see an example of what this code would look like inside your models.py file!

```python
from django.contrib.auth.models import User

# Create your models here.
class UserProfileInfo(models.Model):

    # Create relationship (don't inherit from User!)
    user = models.OneToOneField(User)

    # Add any additional attributes you want
    portfolio = models.URLField(blank=True)
    picture = models.ImageField(upload_to='profile_pics')

    def __str__(self):
        # Built-in attribute of django.contrib.auth.models.User !
        return self.user.username
```

Django

- Notice a new field we haven't seen yet, an ImageField.
- This will allow you to store images to a model, typically we will keep any user uploaded content like this in the media file.

Django

- In order to work with images with Python we will need to install the Python Imaging Library with:
    - **pip install pillow**

Django

- Some users may get an error on this command indicating something like jpeg support disabled, in which case use:

```
pip install pillow --global-option=”build_ext”

--global-option=”--disable-jpeg”
```

Django

- Once you've created this model you'll have to remember to register it in the admin.py file, with something like:
    - `admin.site.register(UserProfileInfo)`

Django

- Typically images, CSS, JS, etc. all go in the static folder of your project, with the STATIC_ROOT variable path defined inside of settings.py
- User uploaded content will go to the media folder, with the MEDIA_ROOT.

Django

- Next we will want to implement a Django form that the User can use to work with the website.
- Let’s show an example of what this would look like inside the forms.py file of your application.

Django

The code inside forms.py

```
from django import forms
from basic_app.models import UserProfileInfo

class UserProfileInfoForm(forms.ModelForm):
    portfolio = forms.URLField(required=False)
    picture = forms.ImageField(required=False)

    class Meta():
        model = UserProfileInfo
        exclude = ('user',)
```

Django

- Let’s now code through the set-up of what we’ve discussed:
    - User Model
    - Media Directory
    - Handling Images
    - User Form

Django

- Once we've done that we can begin to focus on creating a registration view.

Registration

Django Level Five

PIERIAN DATA

Django

- A lot of the coding for working with Users and Authorization happens in the views.py file.
- The basic idea is that we check if there is a POST request and then perform some sort of action based off that information.

Django

- Sometimes we will want to save that information directly to the database.
- Other times, we will set commit=False so we can manipulate the data before saving it to the database.
- This helps prevent collision errors.

Django

- Figuring out the registration views is an extension of what we learn about when discussing Django Forms.
- Make sure to review that content if you don’t remember it!

Django

- This entire process is best shown through code, so let’s jump to our views.py
- We will also fix a small mistake made in the previous lecture when working with the forms, and how to keep an eye out for those errors! Let’s get started!

Logins

Django Level Five

PIERIAN DATA

Django

- Once a user is registered, we want to make sure that they can log in and out of the site.
- In this lecture we will go through the entire process of creating log in / log out functionality.

Django

- This process involves:
    - Setting up the login views
    - Using built-in decorators for access
    - Adding the LOGIN_URL in settings
    - Creating the login.html
    - Editing the urls.py files