Django Level Two

Time to level up your learning!

Django

- Now that you've reached Level Two, let's quickly review the previous topics:
  - Setting up Projects and Applications
  - Creating Views and Mapping URLs
  - Using simple Templates and tags
  - Serving static media files

Django

- In this lecture we will do a run through of the workflow aspects we've learned about so far to get a quick review.
- We will be using the two project folders from Django Level One for Level Two.

Django

- In Level Two we will begin to discuss Models and Databases and how to use them with Django!
- Then we will also discuss how to use the admin interface!

Let’s get started!

Django - Models

Learn how to use Models and Databases!

Django

- An essential part of any website is the ability to accept information from a user and input it into a database and retrieve information from a database and use it to generate content for the user.

Django

- We use Models to incorporate a database into a Django Project.
- Django comes equipped with SQLite.
- SQLite will work for our simple examples, but Django can connect to a variety of SQL engine backends!

Django

- In the settings.py file you can edit the ENGINE parameter used for DATABASES
- To create an actual model, we use a class structure inside of the relevant applications models.py file

Django

- This class object will be a subclass of Django’s built-in class:
    - django.db.models.Model
- Then each attribute of the class represents a field, which is just like a column name with constraints in SQL

Django

- This will all feel very natural if you have some SQL experience, but in case you don’t let’s quickly review what a SQL database is like!

Django

- SQL operates like a giant table, with each column representing a field, and each row representing an entry.

WebsiteId	WebSiteName	URL
1	Google	www.google.com
2	Facebook	www.facebook.com

Django

- Each column has a type of field, such as a CharField, IntegerField, DateField, etc.
- Each field can also have constraints

WebsiteId	WebSiteName	URL
1	Google	www.google.com
2	Facebook	www.facebook.com

Django

- For example, a CharField should have a max_length constraint, indicating the maximum number of characters allowed

WebsiteId	WebSiteName	URL
1	Google	www.google.com
2	Facebook	www.facebook.com

Django

- The last thing to note is table (or models) relationships.
- Often models will reference each other

WebsiteId	WebSiteName	URL
1	Google	www.google.com
2	Facebook	www.facebook.com

Django

- For this referencing to work we use the concepts of Foreign Keys and Primary Keys.

WebsiteId	WebSiteName	URL
1	Google	www.google.com
2	Facebook	www.facebook.com

Django

- Imagine we now have two models.
- One to store website information, another to store date information

WebsiteId	WebSiteName	URL
1	Google	www.google.com
2	Facebook	www.facebook.com

WebsiteId	Date Accessed
1	2018-01-01
2	2018-02-03

Django

- We could say that the WebsiteId column is a primary key in the left table and foreign key in the right table

WebsiteId	WebSiteName	URL
1	Google	www.google.com
2	Facebook	www.facebook.com

WebsiteId	Date Accessed
1	2018-01-01
2	2018-02-03

Django

- A primary key is a unique identifier for each row in a table

WebsiteId	WebSiteName	URL
1	Google	www.google.com
2	Facebook	www.facebook.com

WebsiteId	Date Accessed
1	2018-01-01
2	2018-02-03

Django

- A foreign key just denotes that the column coincides with a primary key of another table

WebsiteId	WebSiteName	URL
1	Google	www.google.com
2	Facebook	www.facebook.com

WebsiteId	Date Accessed
1	2018-01-01
2	2018-02-03

Django

- Later on we will move on to discuss One-to-one or Many-to-many relationships

WebsiteId	WebSiteName	URL
1	Google	www.google.com
2	Facebook	www.facebook.com

WebsiteId	Date Accessed
1	2018-01-01
2	2018-02-03

Django

- Later on we will move on to discuss One-to-one or Many-to-many relationships

WebsiteId	WebSiteName	URL
1	Google	www.google.com
2	Facebook	www.facebook.com

WebsiteId	Date Accessed
1	2018-01-01
2	2018-02-03

Django

- That should be enough for our understanding of models in Django
- Now let’s show an example of the models class that would go into the models.py file of your application

Django

```
class Topic(models.Model):
    top_name = models.CharField(max_length=264, unique=True)

class Webpage(models.Model):
    category = models.ForeignKey(Topic)
    name = models.CharField(max_length=264)
    url = models.URLField()
```

Django

```
class Webpage(models.Model):
    topic = models.ForeignKey(Topic)
    name = models.CharField(max_length=264)
    url = models.URLField()

    def __str__(self):
        return self.name
```

Django

- After we've set up the models we can migrate the database
- This basically let's Django do the heavy lifting of creating SQL databases that correspond to the models we created

Django

- Django can do this entire process with a simple command:
  - `python manage.py migrate`
- Then register the changes to your app, shown here with some generic “app1”:
  - `python manage.py makemigrations app1`

Django

- Then migrate the database one more time:
  - `python manage.py migrate`
  - We can then later on use the shell from the manage.py file to play around with the models

Django

- In order to use the more convenient Admin interface with the models however, we need to register them to our application’s admin.py file.

Django

- We can do this with this code:
  - ```
    from django.contrib import admin
     from app.models import Model1,Model2
    admin.site.register(Model1)
    admin.site.register(Model2)
    ```
    ```

Django

- Then with the models and database created, we can use Django's fantastic Admin interface to interact with the database.
- This Admin interface is one of the key features of Django!

Django

- In order to fully use the database and the Admin, we will need to create a “superuser”
- We can do this with the following:
    - `python manage.py createsuperuser`

Django

- In order to fully use the database and the Admin, we will need to create a “superuser”
- Providing a name, email, and password

Django

- Once we've set up the models, it's always good idea to populate them with some test data
- We will use a library to help with this called Faker and create a script to do this.

Django

- Okay, we've discussed a lot already!
- In the next lecture we will code through an example of all of this to help your understanding!

Creating Models

Django Level Two

Django

- We covered a lot of concepts in the previous lecture, so let’s implement them!
- We will continue working with the two project folders from Django Level One, let’s start making some models!

Populating Models

Django Level Two

Django

- It is usually a good idea to create a script that will populate your models with some “dummy” data.
- Let’s show you how to use the Faker library to create this script!

Django - MTV

Learn about the Models-Templates-Views paradigm!

Django

- Django operates on what is known as Models-Templates-Views
- This is also called “MTV” and encompasses the idea of how to connect everything we’ve talked about so far: models, templates, and views

Django

- There are a few basics steps to achieving the goal of serving dynamic content to a user based off the connection of the models, views , and templates.
- Let’s walk through these.

Django

- First: In the views.py file we import any models that we will need to use.
- Second: Use the view to query the model for data that we will need
- Third: Pass results from the model to the template

Django

- Fourth: Edit the template so that it is ready to accept and display the data from the model.
- Fifth: Map a URL to the view.

Django

- We can practice this methodology by changing what we display on the front index page.
- To begin our understanding of this process we will start by generating a table.

Django

- The table will display all the webpages and access records from the AccessRecord database we created and populated.
- We will use template tagging to connect the model to the html page.

Django

- This entire process will introduce a lot of new things, so don’t be intimidated!
- The template tagging can seem especially confusing at first, don’t worry, just follow along, we will be getting tons of practice with this later on!

Django

- After we walk through all of this with some code, you will have a challenge to practice your basic MTV skills.
- Level Three will focus on expanding this idea of MTV and the Mapping URL step (which we haven’t really dived into yet)

Django

- Alright! Let’s get started!

Django Level Two Project Exercise

Django

- We’ve learned a lot about setting up Models-Templates-Views
- It’s time for you to get some practice!
- We will be using the same ProTwo from Django Level One

Django

- Here is what you have to do:
  - Add a new model called User
  - It should have these fields:
    - First Name
    - Last Name
    - Email

Django

- Make sure to make the migrations!
- Then create a script that will populate your database with fake Users.
- Then confirm the populating through the Admin interface.

Django

- Then create a view for your website for the domain extension /users
- This /users page will be an HTML list of the user names and emails
- You will use template tags to generate this content from the User model.

Django

- Remember to change your urls.py files to deal with the changes to the /users extension!
- Let’s get a quick look at what the final page should look like!