Understanding Django's MVT

Group Session

Models/Views/Templates

What is Django?

Django is a high level **web framework**. It is used to help develop secure and maintainable websites. It is a Python based framework.

The primary goal of Django is to ease the creation of complex **database-driven websites.** It helps eliminate repetitive tasks and makes the development process an easy task. It also **takes less time** to build an application after collecting client requirements.

Django is designed to automatically handle many configurations to help us focus on the application development side only.

code institute

Features of Django

- **Security** - provides a secure way to manage passwords (Keeps info private)
- **Versatility** - can be used to deliver content in many formats (eg: HTML/JSON), can provide various different functionalities such as databases and templating engines.
- **Maintainable** - uses the DRY principle - avoids unnecessary code and repetition
- **Fully loaded** - handles site maps, user authentication (allauth), content administration
- **Portability** - well supported by many web hosting sites
- **Scalability** - provides clear separation between each component meaning it can be scaled up for increased traffic at any level with added hardware

MVT Django

MVT - Model/View/Template:

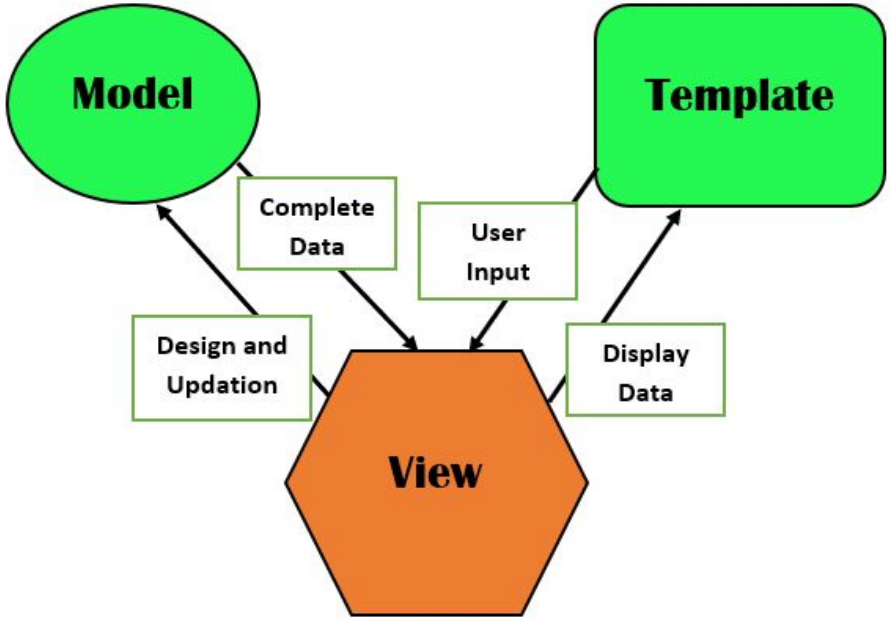
- **Model in Django:** Models are Python objects that define the structure of an application's data and provide mechanisms for managing (adding, modifying, deleting) and querying database records. Aids in database management.

- **Views in Django:** Views in Django serve as a bridge between Model data and Templates. Views in Django MVT, like controllers in MVC, handle all business logic for the web app. It serves as a link between Models and Templates. It receives the user request, retrieves relevant data from the database, and renders the template along with the retrieved data. Views convey data and renders a template while executing business logic and interacting with a model.

- **Templates in Django:** Django uses templates like View in MVC. Templates are entirely in charge of the User Interface. It handles all of the web page's static elements, and the HTML visitors will see.Takes care of User Interface.

MVT Django

MVT - Model/View/Template:



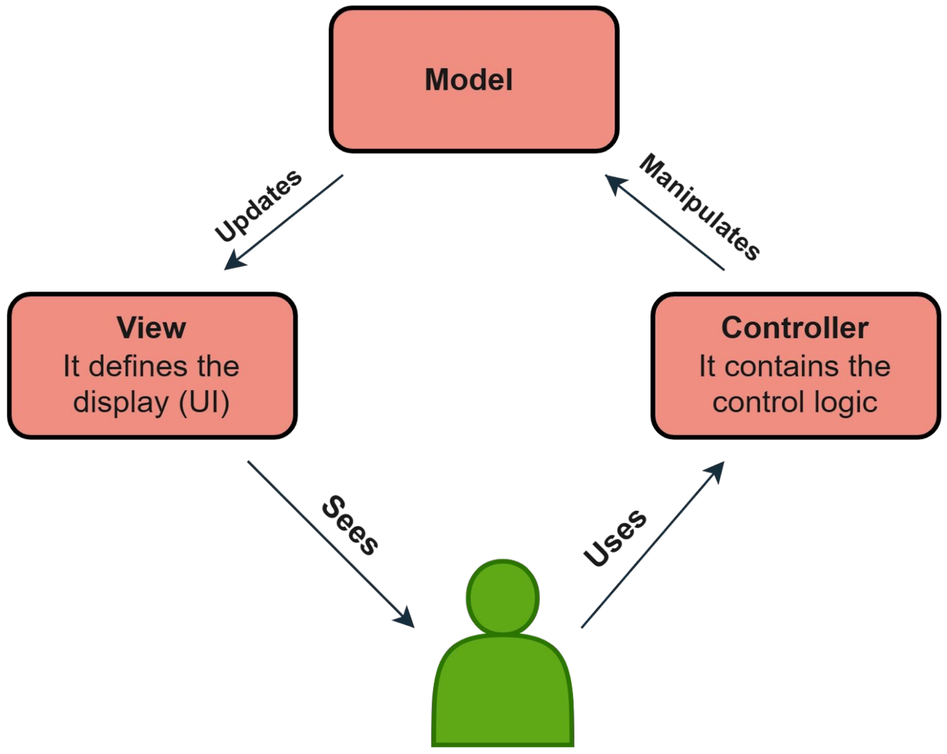
MVC Multi-Language Use

MVC - Model/View/Controller:

- **Model:** A model serves as an interface for data stored in a database. It is in charge of data maintenance and handling the logical data structure for the entire web application.
- **Views:** A view is a user interface - displays and gathers Model Data to/from the user Differs from View in Django
- **Templates:** In MVC, a controller is in charge of the entire web application logic. The controller sees the request when a user uses a view, sends an HTTP request, and responds appropriately.

MVC Multi-Language Use

MVC - Model/View/Controller:



How Django Works:

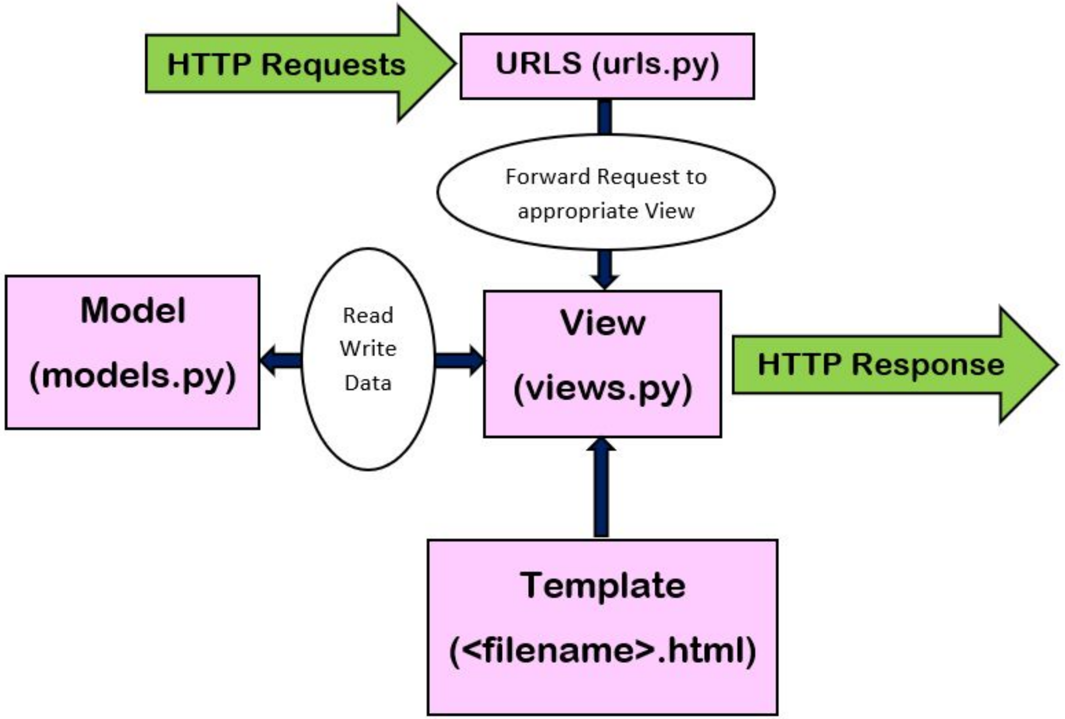
A web application in a traditional data-driven website waits for HTTP requests from the web browser (or other clients). When a request is received, the application determines what is required based on the URL and possibly data in POST or GET methods. Depending on what is needed, it may read or write data from a database or perform other tasks to fulfill the request.

The application will then respond to the web browser by inserting the retrieved data into placeholders in an HTML template and dynamically creating an HTML page for the browser to display.

Django web applications usually separate the code for each step into separate files.
URLs/Models/Templates/Views

The next diagram explains the Django Framework methodology.

Working of Django



Working of Django

As seen in the previous diagram, the steps involved in the working are:

1. Django **receives a URL request** for a resource from the user.
2. The Django framework then looks for the **URL resource**.
3. **If** the URL path **points to a View**, that View is invoked.
4. Now the View will **interact** with the Model to **retrieve the necessary data** from the database.
5. The View then **returns** the user an appropriate template and the retrieved data.

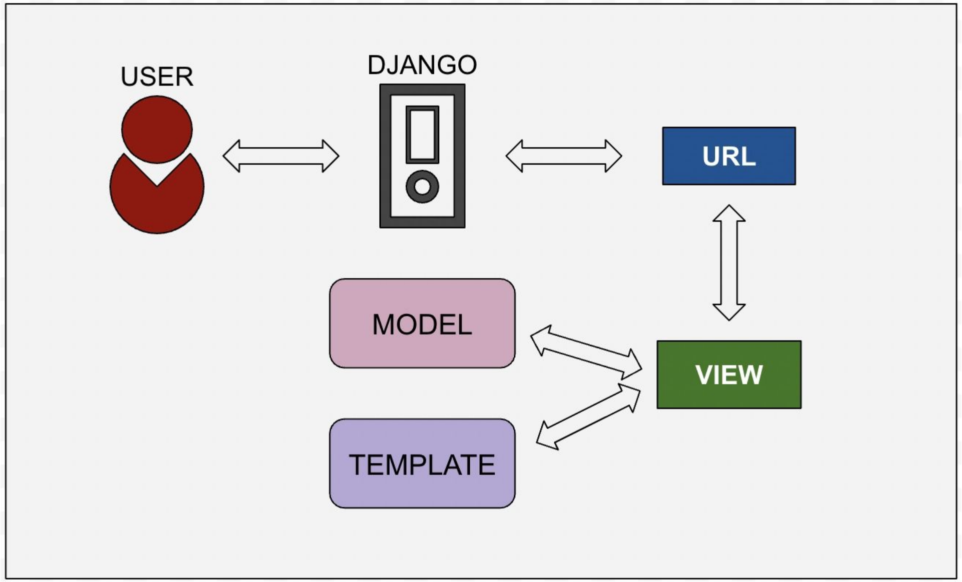
How do we determine the version of Django installed on our system?
To find out what version of Django is installed on the system, open a command prompt and type the following command:

```
python -m django --version
```

Control Flow of Django project with MVT

- A user requests a resource from Django, and Django acts as a controller, checking the URL for available resources.
- A view is created that interacts with the model and renders a template if URL mapping is used.
- Django then responds to the user by sending a response template.

Django using MVT - Coding Ninjas



Setting up a Django project

Steps:

Step 1: In the **command line**:

```
django admin startproject django-mvt
cd django-mvt
python3 manage.py startapp homepage
```

Step 2: In **settings.py** file

```
INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
 'homepage',
]
```

Explanation:

Step 1: In the command line:
This will create an app called homepage

Step 2: In settings.py file, this is to add the name of the new app (‘homepage’) to the INSTALLED_APPS list

Design the Django model

Steps:

Step 3: homepage/models.py

```
from django.db import models

# Create your models here.

class Student(models.Model):
  name = models.CharField(max_length=250)
  rollno = models.CharField(max_length=250)

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

Step 4: Setting Migrations

```
python3 manage.py makemigrations homepage
```

Explanation:

Step 3: A model is simply a database design. This is given to us in our freshly created database (step 1- cd django-mvt).

Step 4: Django uses the sqlite3 database, as defined in the settings

```
DATABASES = {
'default': {
     'ENGINE': 'django.db.backends.sqlite3',
     'NAME': BASE_DIR / 'db.sqlite3',
 }
   }
```

- makemigrations informs Django that changes are to be made to a specific model

Design the Django model

Steps:

Step 5: Running the migrate

```
python3 manage.py migrate
```

***Step 6: Entering the Data in Shell**

```
python3 manage.py shell
```

Explanation:

Step 5: To make changes in the database

***Step 6:** We can read and enter data once inside the shell in the command line

Not currently in CI content?

Accessing the server and Admin Interface

Django features its own admin interface to be used by employees, managers etc to manage the sites data.
This interface is kept private from site visitors to protect the data. eg:(users/emails/addresses)

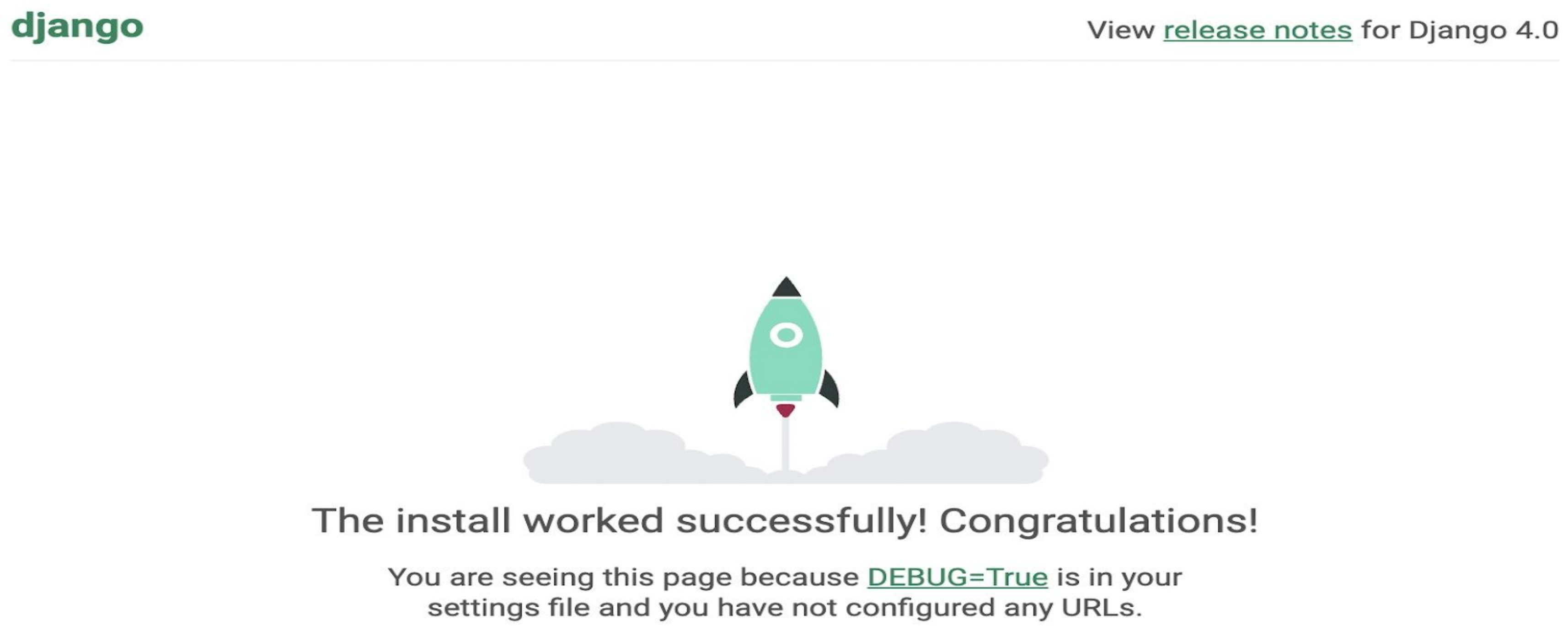
We can give access to the interface using the below command:

```
python3 manage.py createsuperuser
```

Then when prompted enter a username/password/password(again).
Superuser successfully created.

```
python3 manage.py runserver
```

Use the above command to run the server



django

View release notes for Django 4.0

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Admin Login

When you now go to your site eg: http://128.00.0980/admin you will be greeted with a Django administration page that will request the superuser username and password you created.
In here users will have access to the site data such as email/addresses and all models/apps you create.

To register a model with the admin interface,
In **homepage/admin.py:**

```
from django.contrib import admin
from .models import Student
# Register your models here.

    admin.site.register(Student)
```

Server will refresh and add Student to the Django administration. Now you can add new data to the student field by clicking on it.

Templates

The presentation layer in the Django MTV model is the template. This layer communicates with the user, routes requests to the views, and responds to the user.

Django templates are concerned with HTML/CSS/JS. Templates handle all rendering.
Django includes a templating language and the basic HTML/CSS/JS constructs to render dynamic data in the form of templates.

We all know that HTML is a static language and can't interpret the python code written in it as python is a dynamic language, so templates act as a bridge between these two and help to form dynamic HTML pages.

Templates in Django contain static parts also as they are written in desired CSS, HTML, Javascript output. With the help of this, the pages can be rendered efficiently, and the process becomes easy.

Templates

```
TEMPLATES = [
   {
       'BACKEND': 'django.template.backends.django.DjangoTemplates',
       'DIRS': [
               os.path.join(BASE_DIR, 'templates'),
               os.path.join(BASE_DIR, 'templates', 'allauth'),
       ],
       'APP_DIRS': True,
       'OPTIONS': {
           'context_processors': [
               'django.template.context_processors.debug',
               'django.template.context_processors.request',   # required by allauth
               'django.contrib.auth.context_processors.auth',
               'django.contrib.messages.context_processors.messages',
               'django.template.context_processors.media',
           ],
       },
   },
]
```

View & URL

A view and a URL mapped to that view are required to render a template. (views.py in our app)

```
# Create your views here
def index(request):
    """ A view to return the index page """

    return render(request, 'home/index.html')
```

Map the URL to render this view: (urls.py in our app)

```
from django.urls import path
from . import views


urlpatterns = [
    path('', views.index, name='home'),
    path('store/', views.store, name='store'),
]
```

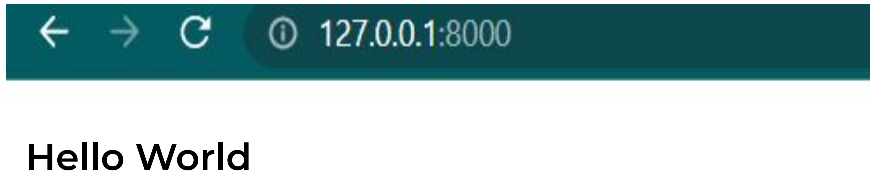
code
institute

Template

Input in index.html:

```
<!doctype html>
<html lang="en">
  <head>
           <title>Arin Beauty</title>
  </head>
  <body>
          <h2>Hello World</h2>
  </body>
</html>
```

Output:


127.0.0.1:8000

Hello World

Django Template Language (Tags)

Tags are python functions that take one or more values and sometimes an optional argument. The process accordingly and return the value.

Tags are represented like such:

```
{% tag_name %}
```

Commonly used tags in Django language

1. Comment
2. Cycle
3 . Extends
4. If
5. For loop
6. Boolean operators

Django Template Language (Filters)

Filters are similar to tags, but the difference is that tags cannot change the value, but they can change the value according to the user's need. Used to transform the value.

Filters are represented like such:

```
{{ variable_name | filter_name }}
```

Commonly used filters

1. Add
2. Center
3. Cut
4. Last
5. Lower
6. slice

Template Inheritance

Template inheritance is an approach to building a base "skeleton" template that contains all the common elements of your site and defines blocks that child templates can override.

Template inheritance is represented like such:

```
{% extends 'template_name.html' %}
```

FAQ

- **What is the use of the ‘manage.py’ file in the Django project using MVT?**

Ans: ‘manage.py’ file is used to interact with our project from the command line (for example, to start the server, sync the database, and so on). In the command window, type the following code to get the complete list of commands that ‘manage.py’ may execute:

```
$ python3 manage.py help
```

- **How is the Django project deployed?**

Ans: The Django project is deployed using the ‘wsgi.py’ file. It helps communication between our Django application and the webserver.