MVC pattern in Django

Christopher Yong · Follow
4 min read · Aug 28, 2021

Introduction

Django is a popular framework, especially used in developing web applications. It's typically known as an MVC(Model View Controller) framework. However, some may debate that it's an MVT(Model View Template) framework.

Source: https://dev.to/nishantwrp/reasons-why-django-is-the-best-web-framework-3f4h

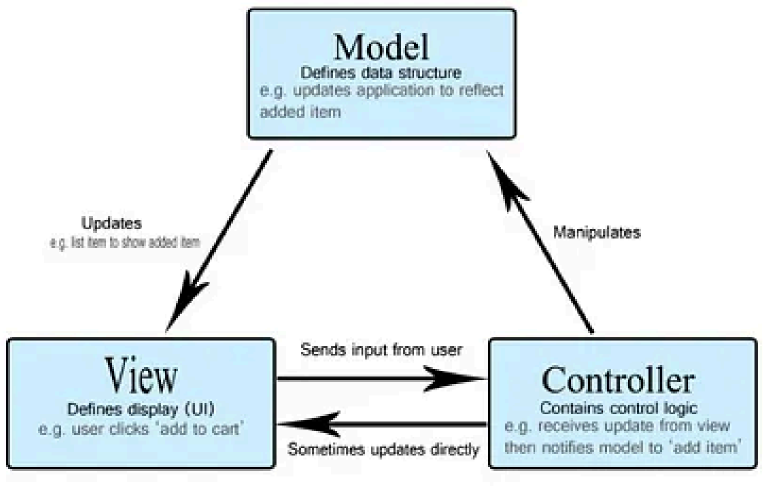
MVC Pattern

First of all, I'll give a brief introduction to the MVC design pattern. MVC consists of three parts:

1. Model: Manage business-logic and data

2. View: Manage the presentation or the layout of the application

3. Controller: Connects Model with View, and routes request.


Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

Source: https://developer.mozilla.org/en-US/docs/Glossary/MVC

MVC Implementation in Django

Next, how does Django adopts and implements the MVC design pattern? We can have a look at the typical file structure of a Django project below from the official Django documentation.

```
polls/
    __init__.py
    admin.py
    apps.py
    migrations/
        __init__.py
    models.py  <----------------------------
    tests.py
    urls.py
    views.py <------------------------------
    templates/
        index.html<--------------------------
```

A Django app will have these highlighted files, models.py, views.py, and template files. The template files are just a bunch of HTML files that act as your frontend. These files are also the files that we will do most of our work on.

Let us look at the models.py file first. Models.py in Django is responsible for defining the database structure and entity relationship.

```
from django.db import models


class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
```

The *Question* class represents an entity. Django built-in ORM(Object Relational Model) will create a database table for it. The attributes of the class, e.g. *question_text* and *pub_date* are the columns of the database table.

Now, we will look at the views.py file. In Django, views.py files are used to connects templates and models.

```
from django.http import HttpResponse
from django.template import loader

from .models import Question


def index(request):
    latest_question_list = Question.objects.order_by('-pub_date')[:5]
    template = loader.get_template('polls/index.html')
    context = {
        'latest_question_list': latest_question_list,
    }
    return HttpResponse(template.render(context, request))
```

There are function-based views and class-based views. The function above is an example of function-based view. When it receives a request, it will query data from the database and does necessary processing. It handles the business logic of the application. Finally, it will return the response of request or in the case of using templates, render HTML file with required data.

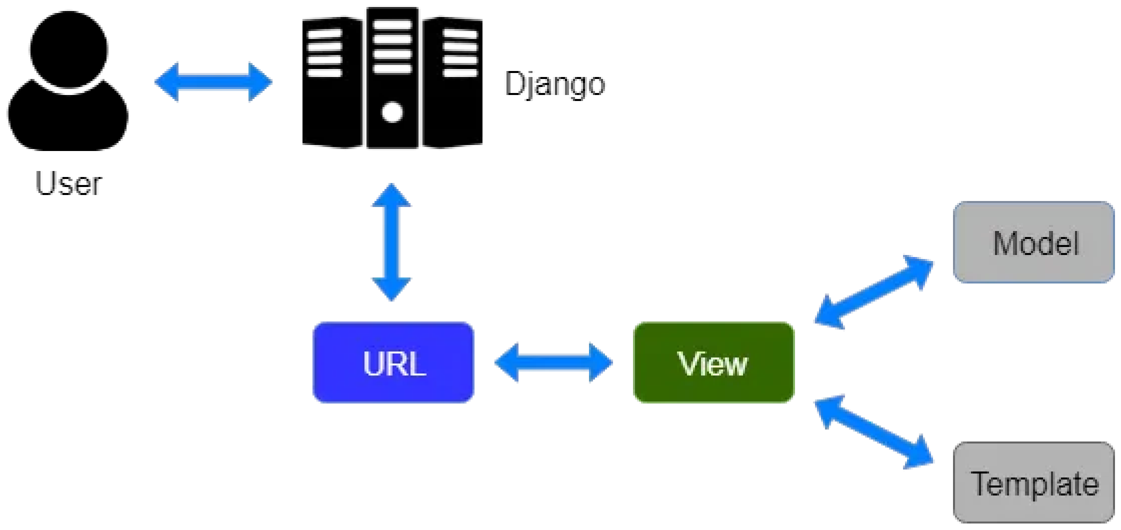
Finally, we will look at the templates, which are ultimately HTML files. These files are the frontend of the application.

```
<h1>{{ question.question_text }}</h1>
<ul>
{% for choice in question.choice_set.all %}
    <li>{{ choice.choice_text }}</li>
{% endfor %}
</ul>
```

One thing special for these HTML files in Django is that. You can directly access the data passed into the HTML file by view using double curly brackets:

```
{{ question.question_text }}
```

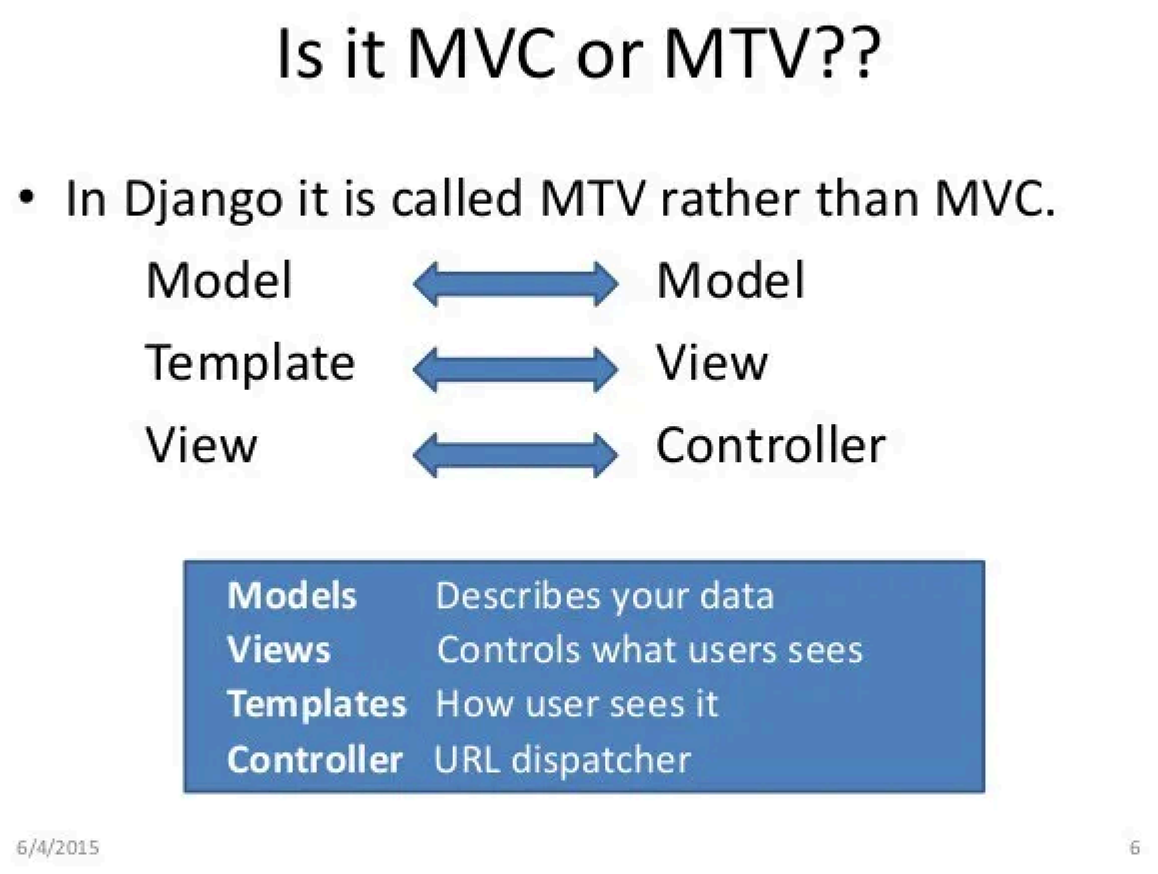
Now, we can take an overview to look at how MVC is implemented in Django. When an URL is visited, the view function will receive a request. It will query the database defined in models.py and done all the necessary processing and render the response using template files.



Source: https://www.onlinetutorialspoint.com/django/django-model-view-template-mvt-overview.html

MVC or MTV?

We can now answer the question about is Django using MTV or MVC. Django's view actually acts as the controller function in typical MVC architecture, and the templates have a similar role as view in MVC.



Is it MVC or MTV??

- In Django it is called MTV rather than MVC.

Model	⟺	Model
Template	⟺	View
View	⟺	Controller

Models	Describes your data
Views	Controls what users sees
Templates	How user sees it
Controller	URL dispatcher

6/4/2015 6

Source: https://medium.com/shecodeafrica/understanding-the-mvc-pattern-in-django-edda05b9f43f

As discussed in the official documentation, Django technically is considered as MTV. However, Django developers are concerned about getting things done, instead of the naming convention itself. They interpreted view as which data is presented, instead of what you see. That's why view in Django acts as the controller. So, MTV can be just considered a variation of MVC.

Conclusion

Anyway, MTV keeps the advantage of MVC. It separates the presentation from the business layer. This follows the extremely important principle of separation of concerns. It allows parallel development at different parts. In addition, it promotes loose coupling between the presentation layer and the business logic layer.