Build a Todo Application Using Python & Django 3

Lauvindra Raj

June 2021

1.0 Introduction

Django is a free and open-source web application framework. A framework is a collection of components which help us build sites faster. Django framework have build in pre-written code which we can use to make our websites faster. Django is a back-end framework which allow the user to program at the back-end to enable the end-user to get a response from the server. This is called Client Server Communication. It also has automatic admin interface which supports CRUD which stands for **create,retrieve,update and delete.**

2.0 Objective

- To create a working Todo application using Python and Django 3
- Able to modify and delete user activities

3.0 How to go from Idea to Implementation

Task: Todo App

Decide Features

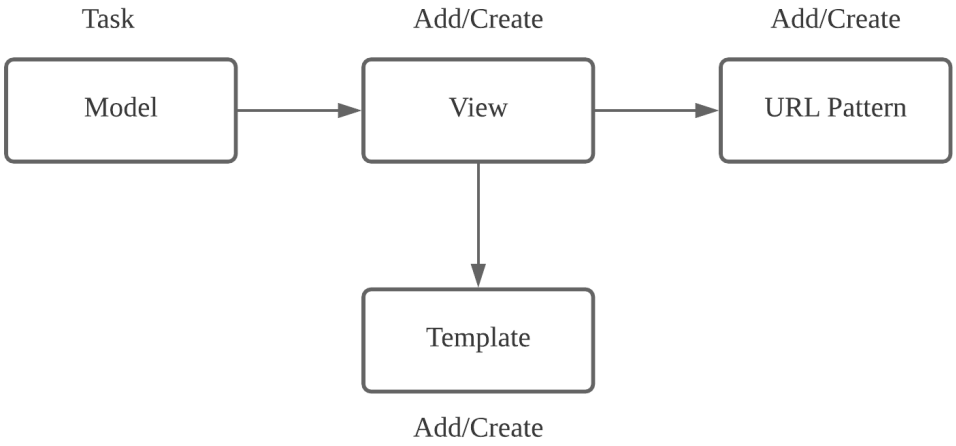
- Add/save tasks
- View tasks
- Update tasks
- Remove tasks

Feature Implementation : Add/save task

The sample action the user might do when using this application and the action should be done to get output to the user

User Input	Actions to get the output
Enters a URL on the browser	Write URL patterns
Visit a page which contains a form	Create a template with a form
Enters task data and click submit	A view which handles the POST request and a model which saves the data

Techniques : Create



Task

- We start by creating models by creating model.py file ,after creating a view we need to create URL pattern by creating urls.py
- Add myapp inside the installed Apps in settings.py directory
- Make migration in terminal to apply the changes made
- Create a superuser to get access to the admin panel using terminal
- Register the models inside admin.py

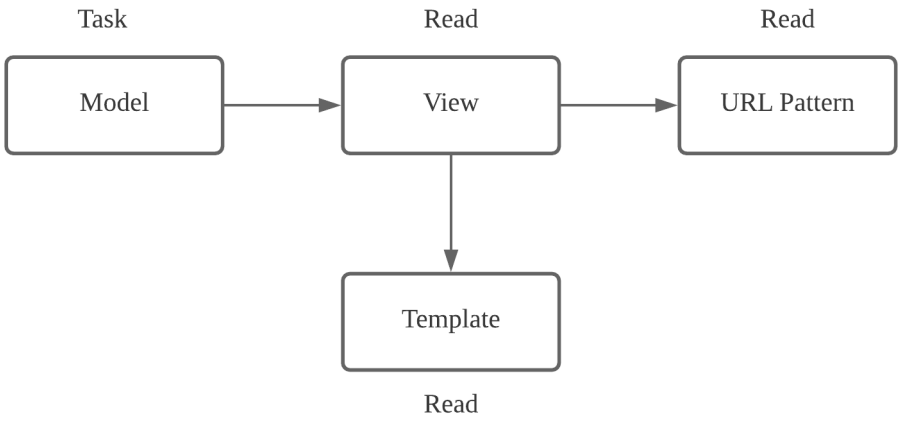
View

- Create a view to render the form template.
- Template which contains a form to accept the data, create a html file, which in here we create insides a template folder.
- In views.py we create tasks to be executed.

Template

- Connect the view with the URL patterns
- To make the website more attractive bootstrap such as CSS and Forms for classes.
- Extend the bootstrap link inside add.html by using block body and end block.

Techniques : Retrieve



Task

- For for the read functionality, we already had the model template.
- Then we have created a simple view to get all the objects or the tasks from the database.

View

- So for the read functionality, we already had the model template. Then we have created a simple view to get all the objects or the tasks from the database.

Template

- So for the read functionality, we already had the model template. Then we have created a simple view to get all the objects or the tasks from the database. And once we got those tasks, we passed those tasks to this template. Then we have rendered this template.

Homepage URL

- And now in order to actually go ahead and trigger that view, we have connected the view with URL pattern which is the Homepage URL in this case.

Technique : Update

- Use Django model form by create a form.py file in myapp directory
- Mention the field we want to update
- Create views inside the views.py file which will render the edit file that will be created later
- Create an edit file, input, method and submit button.
- To view all of it we update the class inside index.html

Technique : Delete

- Create a view called as delete
- Connect it using urls.py file
- A post request is created after clicking the submit to delete button and it is handle in the views.py
- Link the delete.html file in index.html using href functionality.
- Create a url pattern for the update view.

4.0 Class based generic views

A view is a callable which accepts a request and returns a response. There are two type of views which is function and class based views. Django provides built-in class based generic views.

Generic views are also class based views, which is used for listing out objects, displaying their details, adding objects, updating objects & deleting objects.

Types of generic views are Listview, Detailview, Createview, Updateview, Deleteview. This is where the editable function by the user is created in this project.

The difference between class based view (cbv) and function based views (fbv) are cbv are much more easier to write as they require much less amount of code.