**Django** is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Webframework is collection of tools which are used to build up websites.

**Django provides**

* **Object-relational mapping (ORM)** – which helps make database query
* **URL Routing** – which helps in determining what logic to follow depending upon url of the website
* **HTML templating**
* **Form handling**
* **Unit testing tools**

**Django is NOT**

* **A programming language**- its built upon python programming language to build website
* **A web server** – although or dev it comes with an inbuilt server, but for deployments that server is not usable

**Install django in anaconda**

*conda install -c anaconda django*

**Create project**

Go to the path where you want to create project, then write command

*django-admin.py startproject projectname*

for anaconda:

*python C:\Users\HP\Anaconda3\Scripts\django-admin.py startproject test\_project*

**Files created under project**

* **manage.py** – used to run the various commands under project
* **/\_\_init\_\_.py** – tells python that this folder contains python files
* **/wsgi.py** – provides hook for webservers
* **/settings.py** – configure django
* **/url.py** – routes requests based on url

*Generally not edit first three files*

**Run project**

From project folder write

*python manage.py runserver*

*localhost:8000* – to run on browser

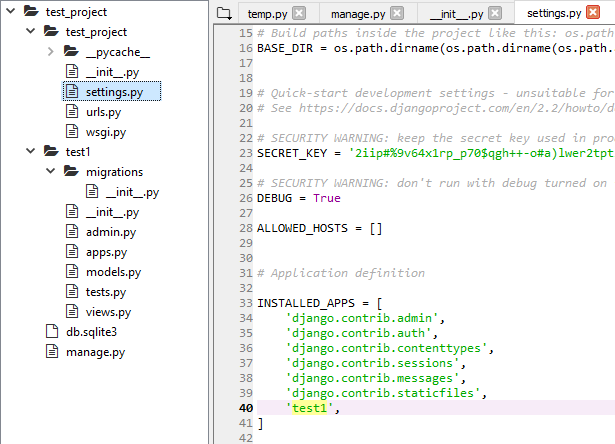
**create django app**

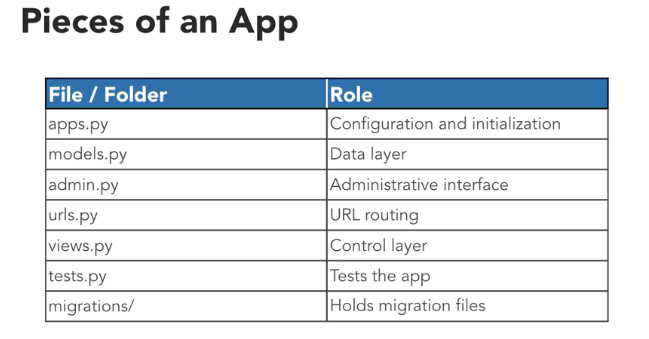
Django app is a component within a django project. An app is a folder with set of python files. Each app supplies a set of related feature for a specific purpose.

*python manage.py startapp test1*

**add django app to project**

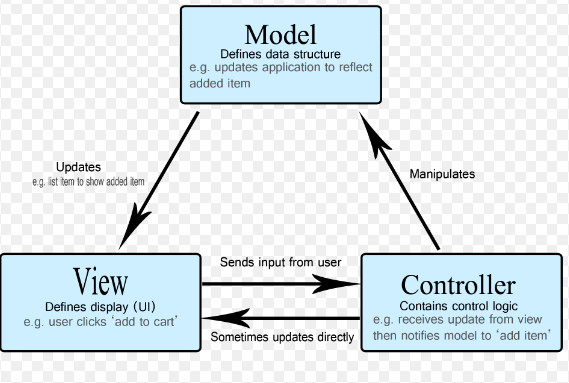
in projects/setting.py add appname in INTSALLED\_APPS section

**



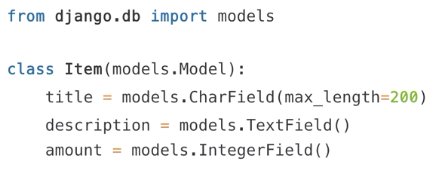
* apps.py- contains setting specific to this app
* models.py- database layer which django uses to construct database schema and query
* views.py –defines the logic and control flow for handling request and define each db response that are returned
* migration.py – used for migration of db as we create and migrate db overtime.

Django uses MVC (model view controller architecture)



**Models** – is a class inheriting from *django.db.model.Model* and is used to define field as class attributes

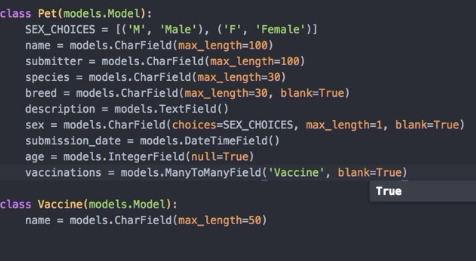
Define fields



Difference between Null=True & Blank=True

Blank=True submission for an integer value result in submission of 0 while for null its null only.

Note that this is different than [null](https://docs.djangoproject.com/en/2.2/ref/models/fields/#django.db.models.Field.null). [null](https://docs.djangoproject.com/en/2.2/ref/models/fields/#django.db.models.Field.null) is purely database-related, whereas [blank](https://docs.djangoproject.com/en/2.2/ref/models/fields/#django.db.models.Field.blank) is validation-related. If a field has blank=True, form validation will allow entry of an empty value. If a field has blank=False, the field will be required.



*SEX\_CHOICES = [('M', 'Male'), ('F', 'Female')]*

*sex = models.CharField(choices=SEX\_CHOICES, max\_length=1, blank=True)*

**Migration** generate scripts to change database structure over time.

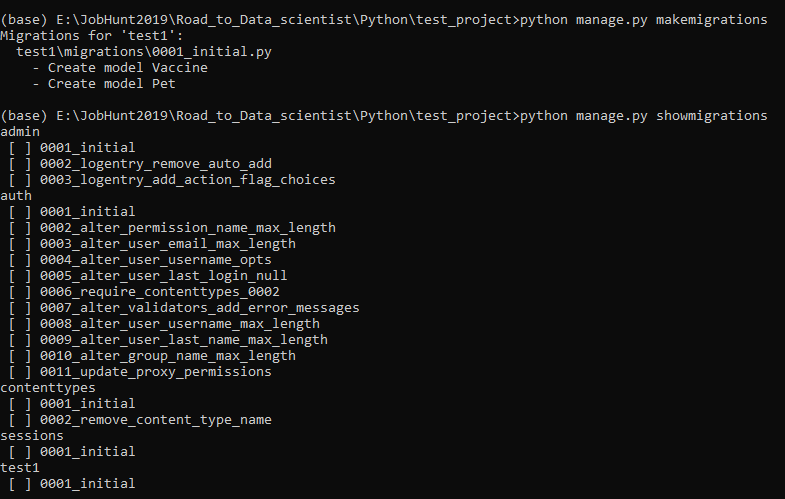
When no tables for model initially then migration is initial migration.

*python manage.py makemigrations*

*python manage.py showmigrations*

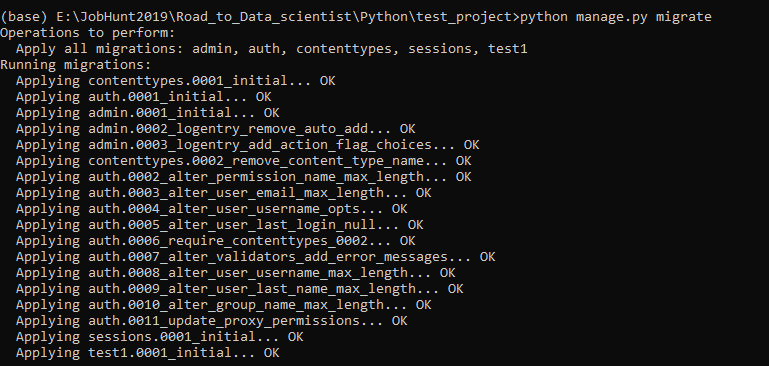
[]- shows not migrated

[X] – migration is applied



applying migrations

*python manage.py migrate*

**

**Create superuser to login for ourselves**

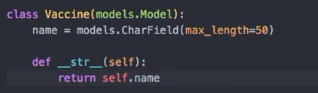
*python manage.py createsuperuser*

<http://localhost:8000/admin/> - access admin page

**

Class of models in admin page and then associate model

For any model instance on interface djnago displays modelname-object, hence need to overwrite the values in model



**Django ORM and queryset**

A QuerySet is, in essence, a list of objects of a given Model. QuerySets allow you to read the data from the database, filter it and order it.

Open django shell - *python manage.py shell*

*from test1.models import Pet*

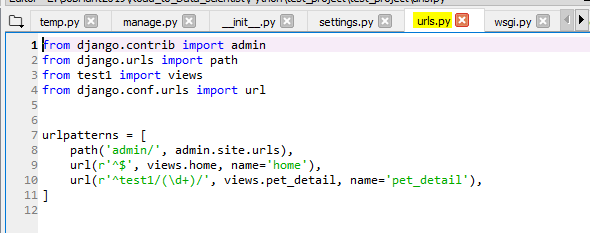
*pets = Pet.objects.all()*

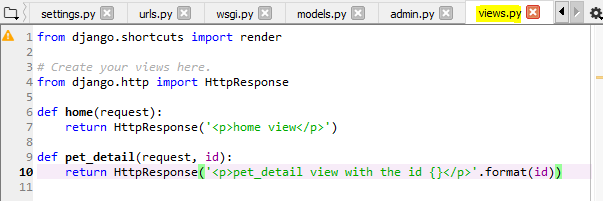
*pet = pets[0]*

*pet.id*

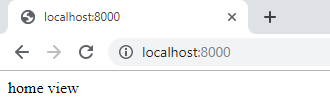
*Pet.objects.get(id=7)*

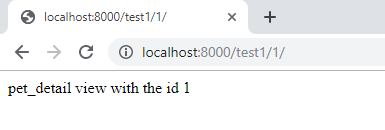
**Implementation of django url patterns**





Strings typically use raw string syntax (r'') so that they can contain sequences like \d without the need to escape the backslash with another backslash



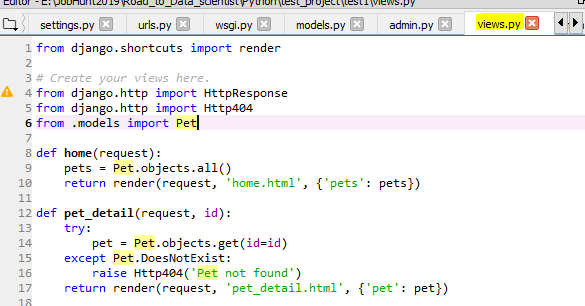
**

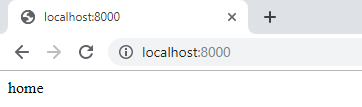
## Implementation of django views

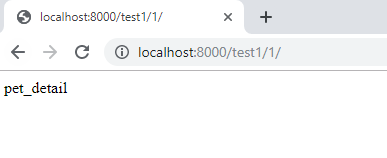
## render()

render(*request*, *template\_name*, *context=None*, *content\_type=None*, *status=None*, *using=None*)

Combines a given template with a given context dictionary and returns an [HttpResponse](https://docs.djangoproject.com/en/2.2/ref/request-response/#django.http.HttpResponse) object with that rendered text.

**

**

**

The **SQLite**library implements a serverless, transactional, self-contained SQL database engine. SQLite is a free database application. The database files that are created by SQLite are given the **.sqlite file** extension and can be used to create customer lists, inventory databases and other database information.

It is a [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) (RDBMS) contained in a [C](https://en.wikipedia.org/wiki/C_(programming_language)) [library](https://en.wikipedia.org/wiki/Library_(computer_science)). In contrast to many other database management systems, SQLite is not a [client–server](https://en.wikipedia.org/wiki/Client%E2%80%93server) database engine. Rather, it is embedded into the end program.

*DB Browser for SQLite* (DB4S) is a high quality, visual, open source tool to create, design, and edit database files compatible with SQLite.

**TBL**

* Load data from csv sqlite- django
* OOPs concepts in django
* def \_\_str\_\_(self):