URL Dispatcher

To design URLs for app, you create a Python module informally named urls.py This module is pure Python code and is a mapping between URL path expressions to view functions.

This mapping can be as short or as long as needed.

It can reference other mappings.

It's pure Python code so it can be constructed dynamically.

urls.py

```
urlpatterns = [
      path(route, view, kwargs=None, name=None)
     ]
```

urls.py

```
urlpatterns = [
      path('learndj/', views.learn_django),
]
```

path ( )

path(route, view, kwargs=None, name=None) - It returns an element for inclusion in urlpatterns.

Where,

- The route argument should be a string or gettext_lazy() that contains a URL pattern. The string may contain angle brackets e.g. <username> to capture part of the URL and send it as a keyword argument to the view. The angle brackets may include a converter specification like the int part of <int:id> which limits the characters matched and may also change the type of the variable passed to the view. For example, <int:id> matches a string of decimal digits and converts the value to an int.
- The view argument is a view function or the result of as_view() for class-based views. It can also be an django.urls.**include()**.
- The kwargs argument allows you to pass additional arguments to the view function or method. It should be a dictionary.
- name is used to perform URL reversing.

path ( )

urls.py

```
urlpatterns = [
      path(route, view, kwargs=None, name=None)
     ]
```

urls.py

```
urlpatterns = [
     path('learndj/', views.learn_Django, {'check': 'OK'}, name='learn_django'),
     ]
```

URL Pattern inside Project

urls.py file is used to define url pattern attached with application or view of application or view function of application.

urls.py file is located inside *inner project folder* not inside *application folder* which means we define url at project level for applications. Defined url name will be used by application user to get response from the application or view function of application.

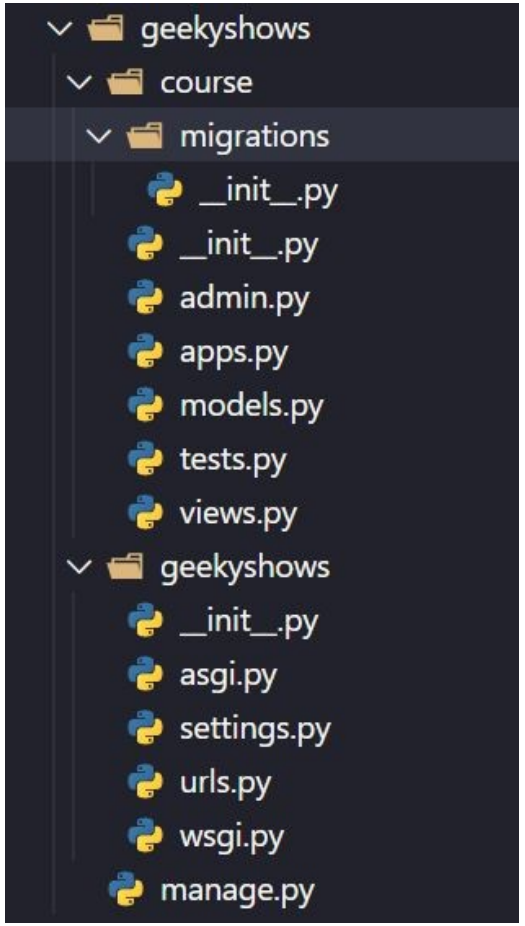
Steps:-

- Open urls.py
- Import Module (Python file) of the application
- Write URL Name and Map it with function

```
urlpatterns = [
        path(route, view, kwargs=None, name=None)
]
```

```
from course import views
urlpatterns = [
        path('learndj/', views.learn_django),
]
```

learndj is mapped with learn_django function which is inside views.py file.



http://127.0.0.1:8000/learndj

URL Pattern inside Project

Single Application with Single function.

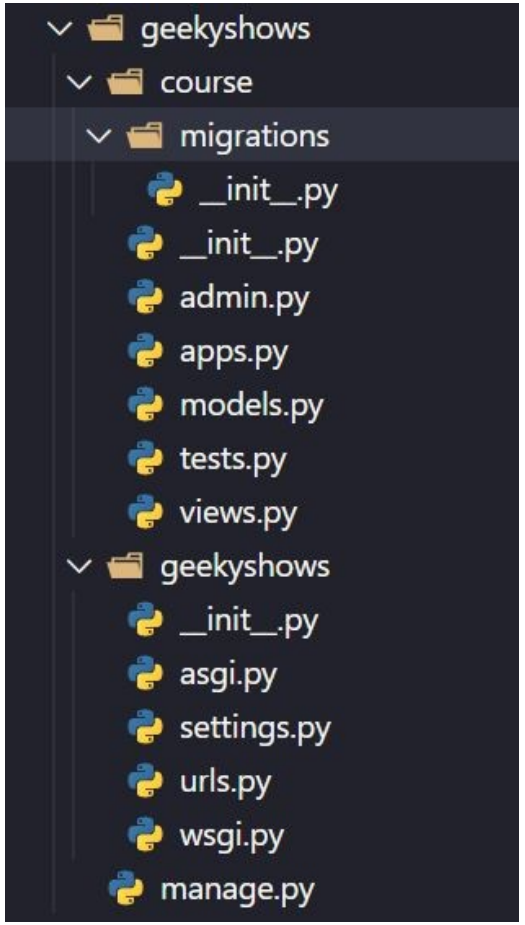
views.py

```
from django.http import HttpResponse
def learn_django(request):
    return HttpResponse('Hello Django')
```

urls.py

```
from course import views
urlpatterns = [
    path('learndj/', views.learn_django),
]
```

http://127.0.0.1:8000/learndj



URL Pattern inside Project

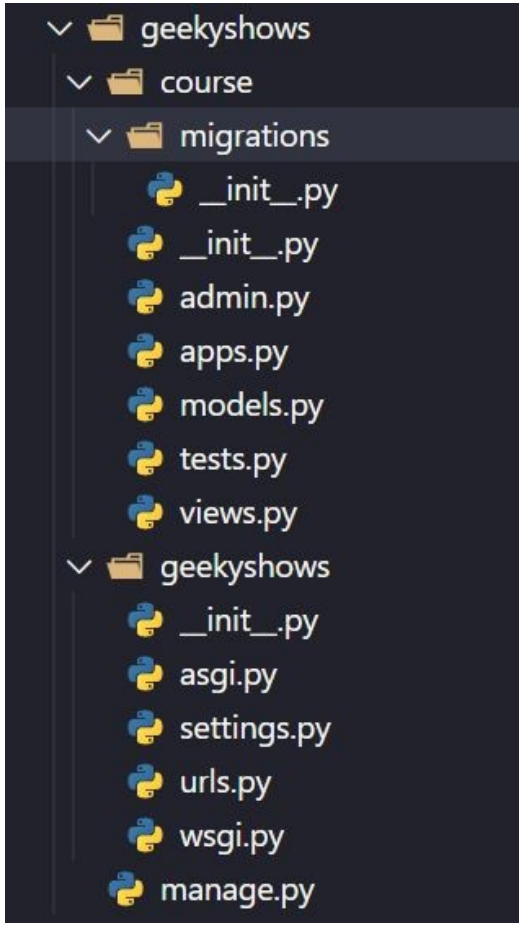
We can define multiple url for one view function. Which means we can access same view function with multiple urls.

urls.py

```
from course import views
urlpatterns = [
        path('learndj/', views.learn_django),
        path('altlearndj/', views.learn_django),
]
```

http://127.0.0.1:8000/learndj

http://127.0.0.1:8000/altlearndj



URL Pattern inside Project

Single Application with multiple functions.

views.py

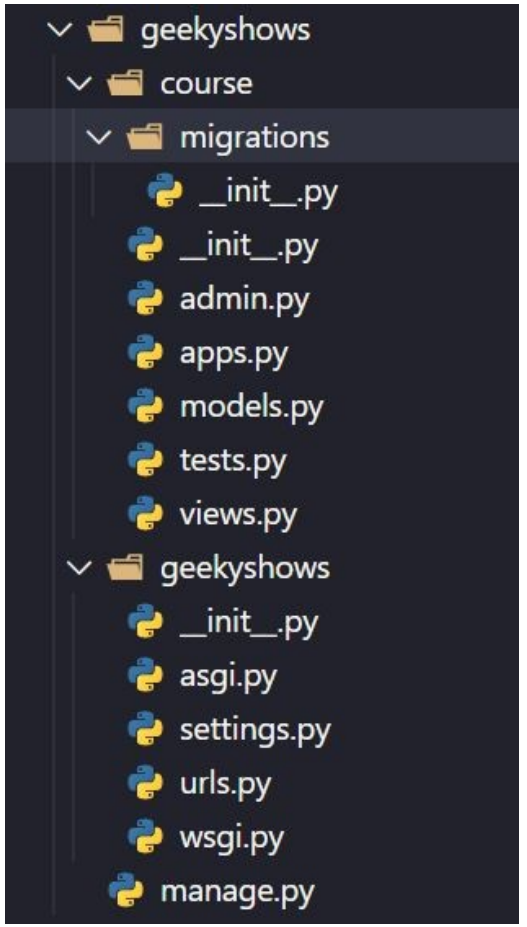
```
from django.http import HttpResponse
def learn_django(request):
    return HttpResponse('Hello Django')
def learn_python(request):
    return HttpResponse('<h1>Hello Python</h1>')
```

urls.py

```
from course import views
urlpatterns = [
    path('learndj/', views.learn_django),
    path('learnpy/', views.learn_python),
]
```

http://127.0.0.1:8000/learndj

http://127.0.0.1:8000/learnpy



Geeky Steps

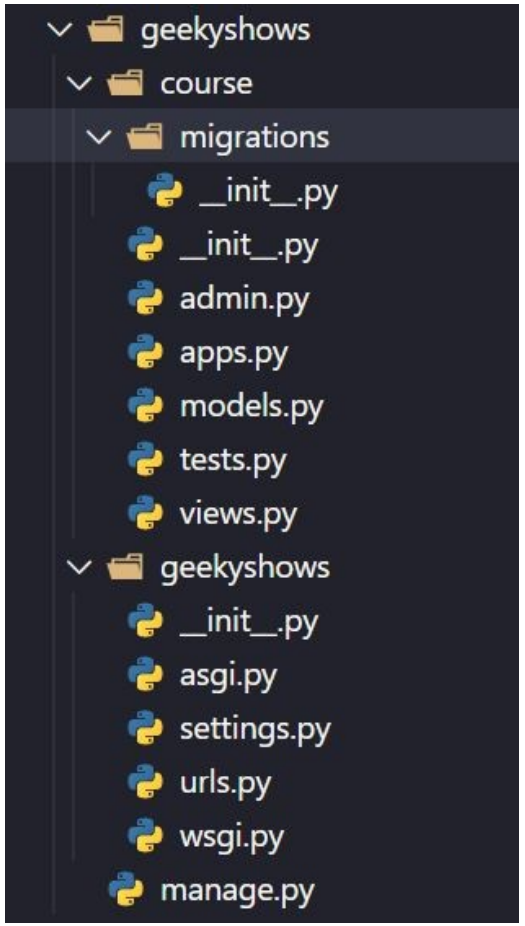
- Create Django Project: *django-admin startproject geekyshows*
- Change Directory to Django Project: *cd geekyshows*
- Create Django Application: *python manage.py startapp course*
- Add/Install Application to Django Project (course to geekyshows) using settings.py file INSTALLED_APPS
- Write View Function inside views.py file
- Define url for view function of application
  - Open ***urls.py***
  - Import views Module of the application

    from course import views
  - Write url Pattern

urlpatterns = [path('learndj/', views.learn_django),

path('learnpy/', views.learn_python),]

  - Save urls.py



How URL Dispatcher Works

- Django determines the root URLconf (urls.py) module to use. Ordinarily, this is the value of the ROOT_URLCONF setting, but if the incoming HttpRequest object has a urlconf attribute (set by middleware), its value will be used in place of the ROOT_URLCONF setting.
- Django loads that Python module and looks for the variable urlpatterns. This should be a sequence of django.urls.path() and/or django.urls.re_path() instances.
- Django runs through each URL pattern, in order, and stops at the first one that matches the requested URL, matching against path_info.
- Once one of the URL patterns matches, Django imports and calls the given view, which is a Python function (or a class-based view).
  - An instance of HttpRequest.
  - If the matched URL pattern contained no named groups, then the matches from the regular expression are provided as positional arguments.
  - The keyword arguments are made up of any named parts matched by the path expression that are provided, overridden by any arguments specified in the optional kwargs argument to django.urls.path() or django.urls.re_path().
  - If no URL pattern matches, or if an exception is raised during any point in this process, Django invokes an appropriate error-handling view.

re_path ( )

re_path(route, view, kwargs=None, name=None) - It returns an element for inclusion in urlpatterns.

Where,

- The route argument should be a string or gettext_lazy() that contains a regular expression compatible with Python's re module. 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. When a match is made, captured groups from the regular expression are passed to the view as named arguments if the groups are named, and as positional arguments otherwise. The values are passed as strings, without any type conversion.
- The view argument is a view function or the result of as_view() for class-based views. It can also be an django.urls.**include()**.
- The kwargs argument allows you to pass additional arguments to the view function or method.
- name is used to perform URL reversing.

re_path ( )

urls.py

```
urlpatterns = [
      re_path(route, view, kwargs=None, name=None)
     ]
```

urls.py

```
urlpatterns = [
     path(r'^learndj/$', views.learn_Django, {'check': 'OK'}, name='learn_django'),
     ]
```