Django

Getting Python in the web

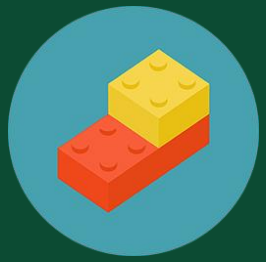
Ahmed Moawad

Course Prerequisites

<HTML
CSS}

Course Objectives

Learn How to use Python in web development.



Learn How Django works and how to build awesome website using it.

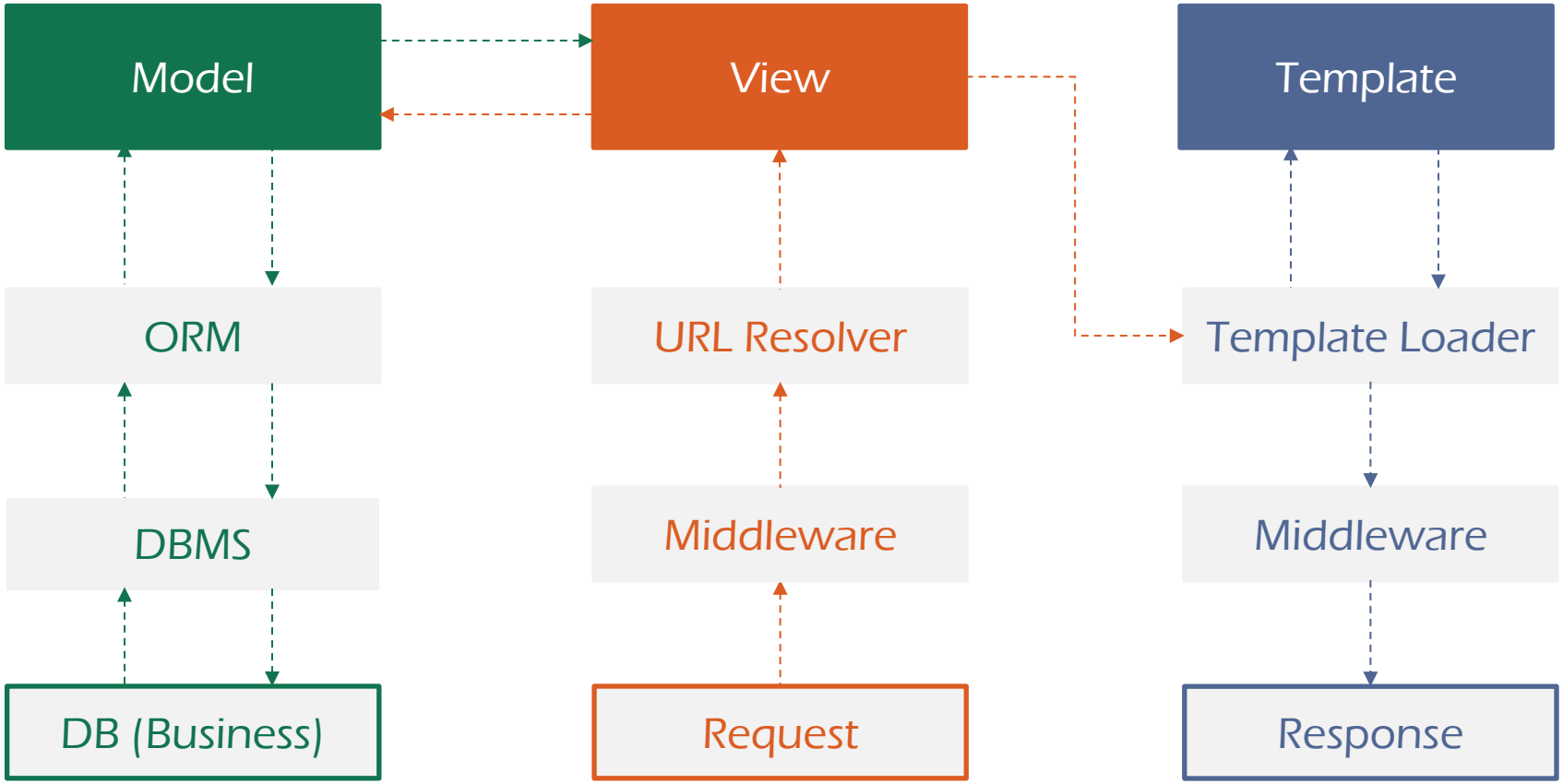
Intro

Intro

Django is a high-level Python Web framework that encourages rapid development .
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.

websites that uses

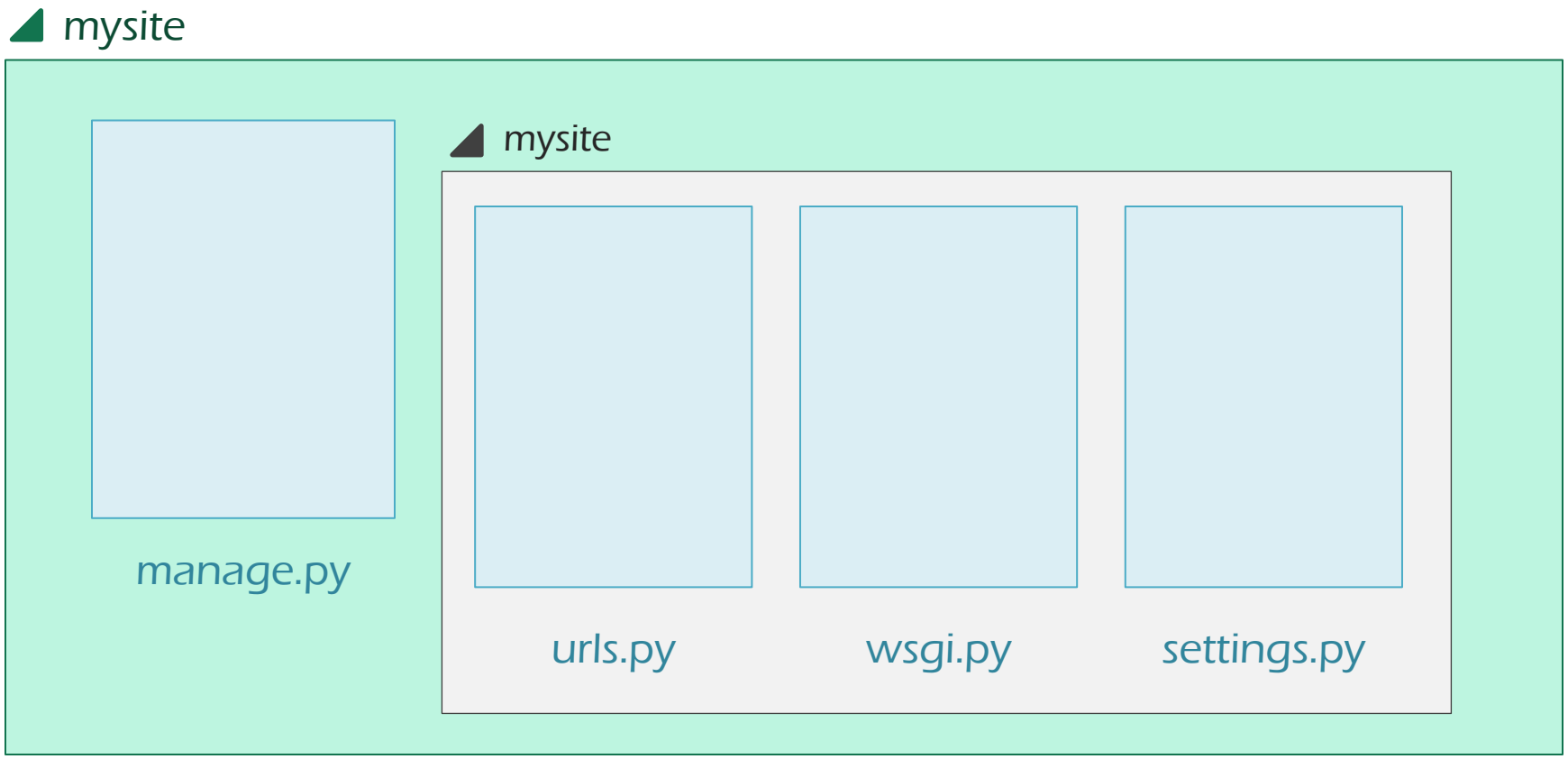
Architecture



First Project

Create

```
$ django-admin startproject mysite
```




```
$ cd mysite
$ python manage.py runserver
```

Settings File

settings.py

is a file that contain the configuration for all your project. It consists of some variables that have a value that affect on the project life style.

`BASE_DIR` Define the base directory to use it in the project.

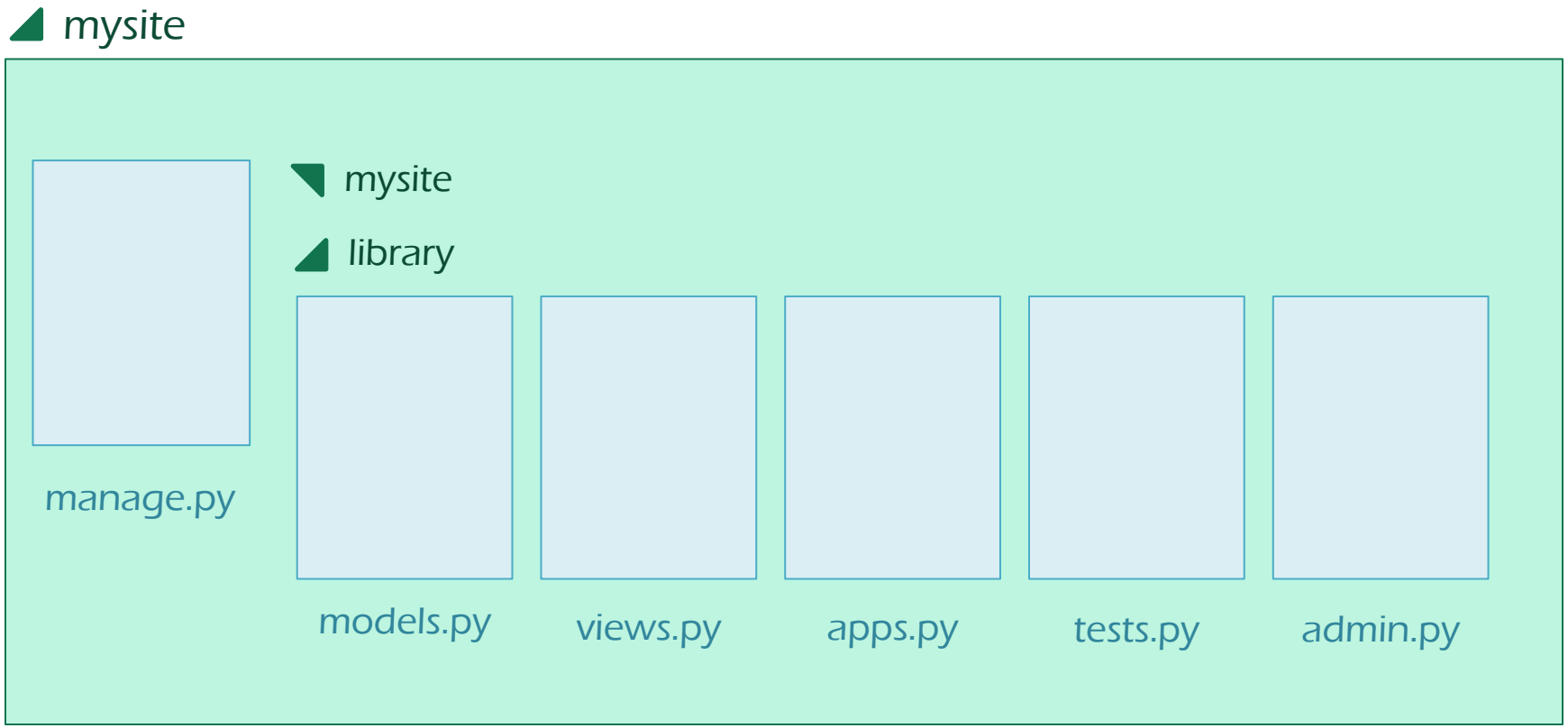
`DEBUG` If True , It directs to that your application in the development mode.

We will introduce other settings later based on the related topics

Apps

Website is a group of web apps

```
$ python manage.py startapp library
```



Make a View

library/views.py

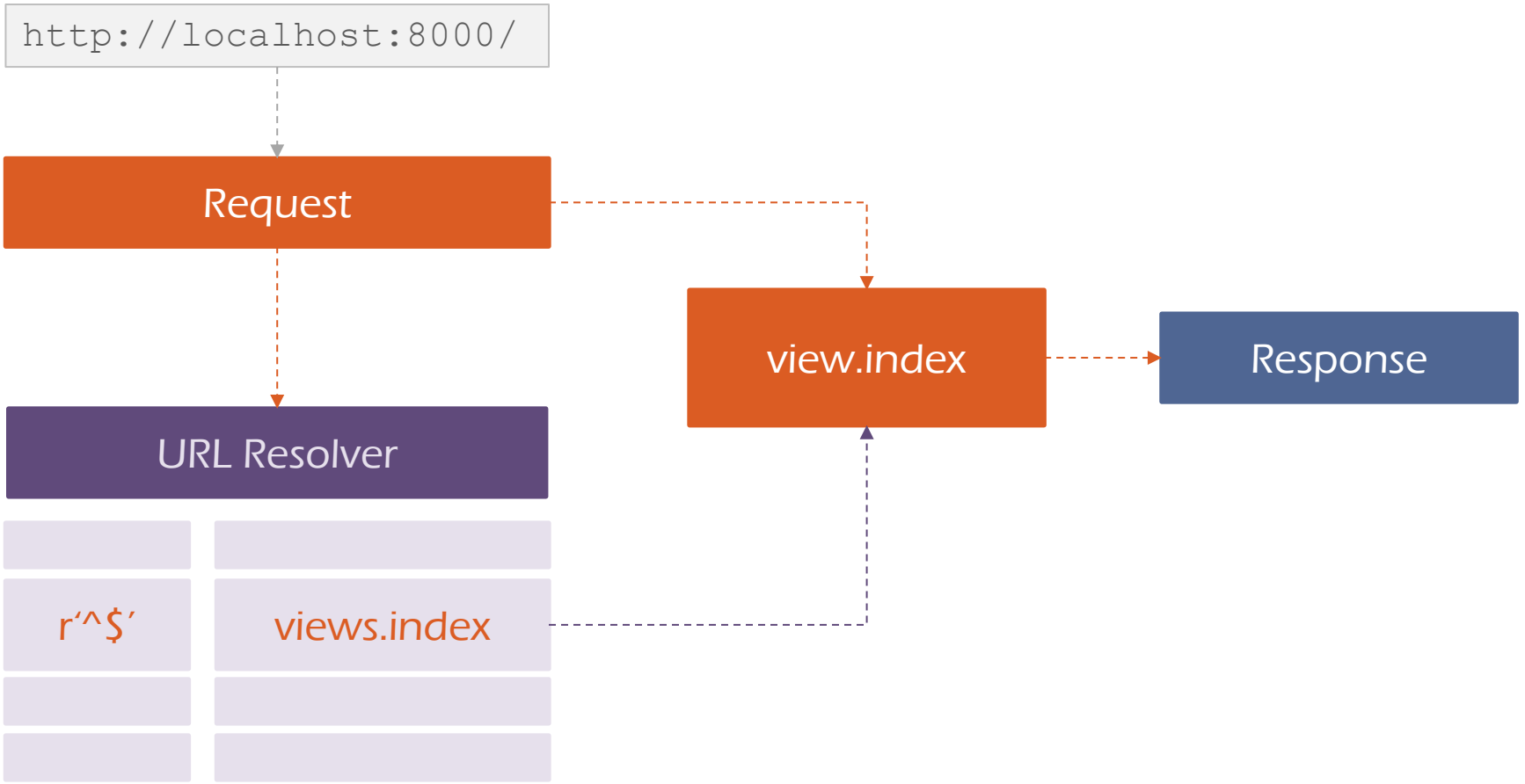
```
from django.http import HttpResponse

def index(request):

        return HttpResponse("Hello World")
```

mysite/urls.py

```
from django.conf.urls import url

from library import views

urlpatterns = [ url(r'^$',views.index) ]
```

Run

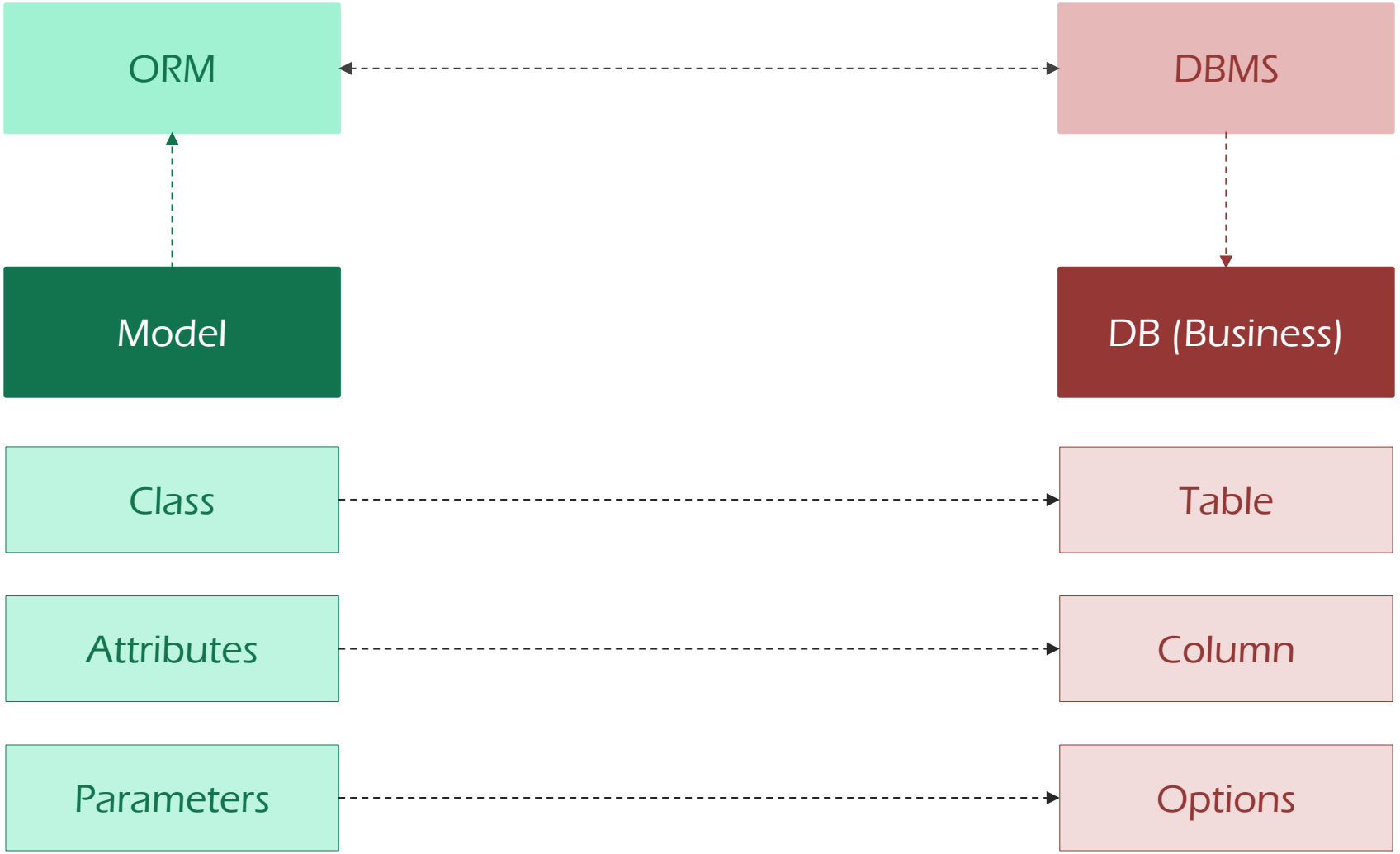
```
$ cd mysite
$ python manage.py runserver
```



Models

Intro

A model is the single, definitive source of information about your data.



Example

models.py

```
from django.db import models

class Book(models.Model):

        title = models.CharField(max_length=100)

        author = models.CharField(max_length=200)

        bid = models.AutoField(primary_key=True)
```

Migrate

makemigrations	sqlmigrate	migrate

mysite/settings.py

```
INSTALLED_APPS = [
        …  # Add library app to the installed apps
        'library.apps.LibraryConfig'
]
```

library/models.py

```
from django.db import models
class Author(models.Model):
        first_name = models.CharField(max_length=200)
        last_name = models.CharField(max_length=200)
```

```
$ cd helloworldapp
$ python manage.py makemigrations
$ python manage.py migrate
```

Settings

mysite/settings.py

```
DATABASES =  {
        'default':{
            #Define the Database Engine
            'ENGINE':'django.db.backends.mysql',
            'NAME': 'bookstore',
            'USER': 'root',
            'PASSWORD': '',
            'HOST': 'localhost',
        }
}
```

Field Types & options

Field options

Field

options

null	If True , It is allowed fro the Field to be null.
choices	A Tuple of choices that Field value can be.
db_column	The name of the database column to use for this field
default	The default value for field if not given
primary_key	If True, Field will be the primary key of the table in DB.
unique	If True, Filed values must be unique.

```
first_name = models.Field(null=False, default="Ahmed", unique=True)
```

Character & Text

`CharField`

A string field, it map the VARCHAR type in SQL.

`EmailField`

A CharField that accepts valid Email addresses only.

`URLField`

A CharField that accepts valid urls only

`options`

`max_length`

`TextField`

A large text field

Numeric & Boolean

`IntegerField`

Values can be integer from -2147483648 to 2147483647.

`AutoField`

An IntegerField that automatically increments according to available IDs.

`DecimalField`

A fixed-precision decimal number, represented in Python by a **Decimal** instance

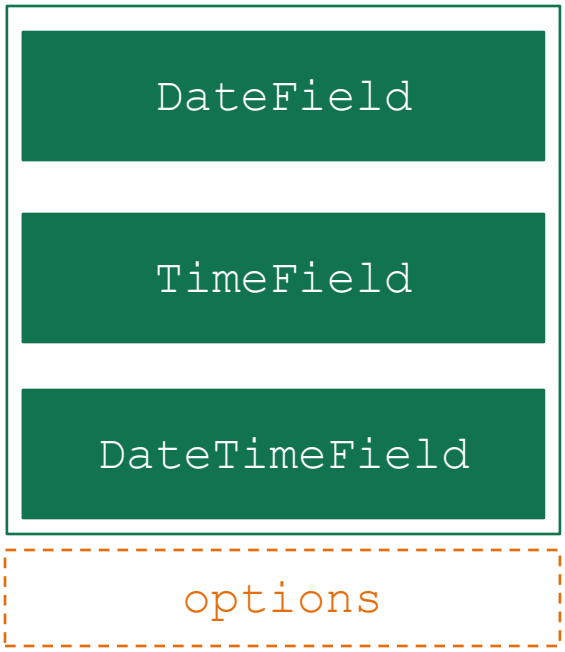
`options`

`max_digits` `Decimal_places`

`BooleanField`

A Field that accept True/False values only.

Date & Time



A date, represented in Python by a datetime.date instance

A date, represented in Python by a datetime.time instance

A date, represented in Python by a datetime.datetime instance

`auto_now`

`auto_now_add`

Relationship Fields

`ForeignKey`	Describe the many-to-one relationship
`ManyToManyField`	Describe the many-to-many relationship
`OneToOneField`	Describe the one-to-one relationship

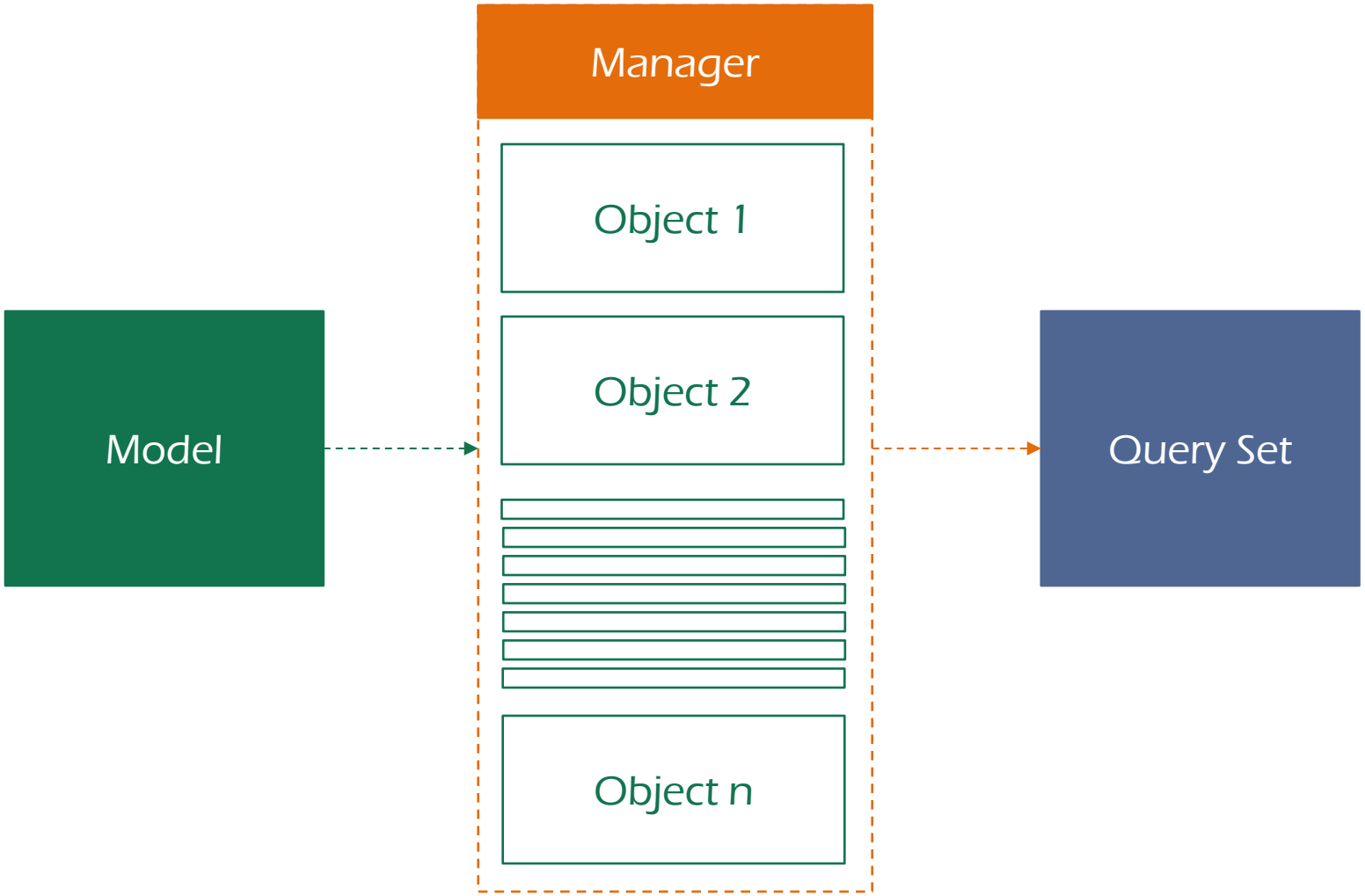
`options` `RelatedModel` `on_delete` `to_field`

models.py

```
from django.db import models

class User(models.Model):

    group_id = models.ForeignKey('Group',on_delete=models.CASCADE)
```

Model Operations

Intro




```
Post . objects . all()
```

Create

INSERT

```
u = User(first_name='Ahmed')
u.last_name= 'Moawad'
u.save()

#or use create method

User.objects.create(first_name='Ahmed', last_name='Moawad')
```

Retrieve Objects

SELECT … WHERE

```
User.objects.create(first_name='Ahmed', last_name='Moawad')

User.objects.create(first_name='Mohamed', last_name='Saeed')

User.objects.create(first_name='Omar', last_name='Saeed', age=12)

User.objects.all()

<QuerySet [<User: User object>, <User: User object>],<User: User object>]>

# To filter the resulting QuerySet based on condition, use filter

User.objects.filter(last_name='Saeed')

<QuerySet [<User: User object>, <User: User object>] >

# To Retrieve single record, use get

User.objects.get(age = 12)

<User: User object>
```

DELETE

```
User.objects.create(first_name='Ahmed', last_name='Moawad')

User.objects.create(first_name='Mohamed', last_name='Saeed')

User.objects.create(first_name='Omar', last_name='Saeed', age=12)

User.objects.filter(last_name='Saeed').delete()
```

UPDATE

```
User.objects.create(first_name='Ahmed', last_name='Moawad')

User.objects.create(first_name='Mohamed', last_name='Saeed')

User.objects.create(first_name='Omar', last_name='Saeed', age=12)

User.objects.filter(last_name='Saeed').update(age = 19)
```

Field Lookups

LIKE

```
User.objects.create(first_name='Ahmed', last_name='Moawad')          #1
User.objects.create(first_name='Mohamed', last_name='Saeed')         #2
User.objects.create(first_name='Omar', last_name='saeed', age=12)    #3
User.objects.filter(first_name__in =['Ahmed', 'Omar'])               [#1,#3]
User.objects.filter(last_name__iexact ='saeed')                      [#2,#3]
User.objects.filter(age__gt =13)                                     [#1,#2]
```

F()

```
from django.db.models import F

User.objects.create(first_name='Ahmed', last_name='Moawad')

User.objects.create(first_name='Mohamed', last_name='Saeed')

User.objects.create(first_name='Omar', last_name='Omar', age=12)

User.objects.filter(last_name= F('first_name'))
```

GROUP BY … HAVING

```
from django.db.models import Avg

User.objects.create(first_name='Ahmed', last_name='Moawad')          #1

User.objects.create(first_name='Mohamed', last_name='Saeed')         #2

User.objects.create(first_name='Omar', last_name='saeed', age=12)    #3

User.objects.count() #3

User.objects.all().aggregate(Avg('age'))

{'age__avg': 14}
```

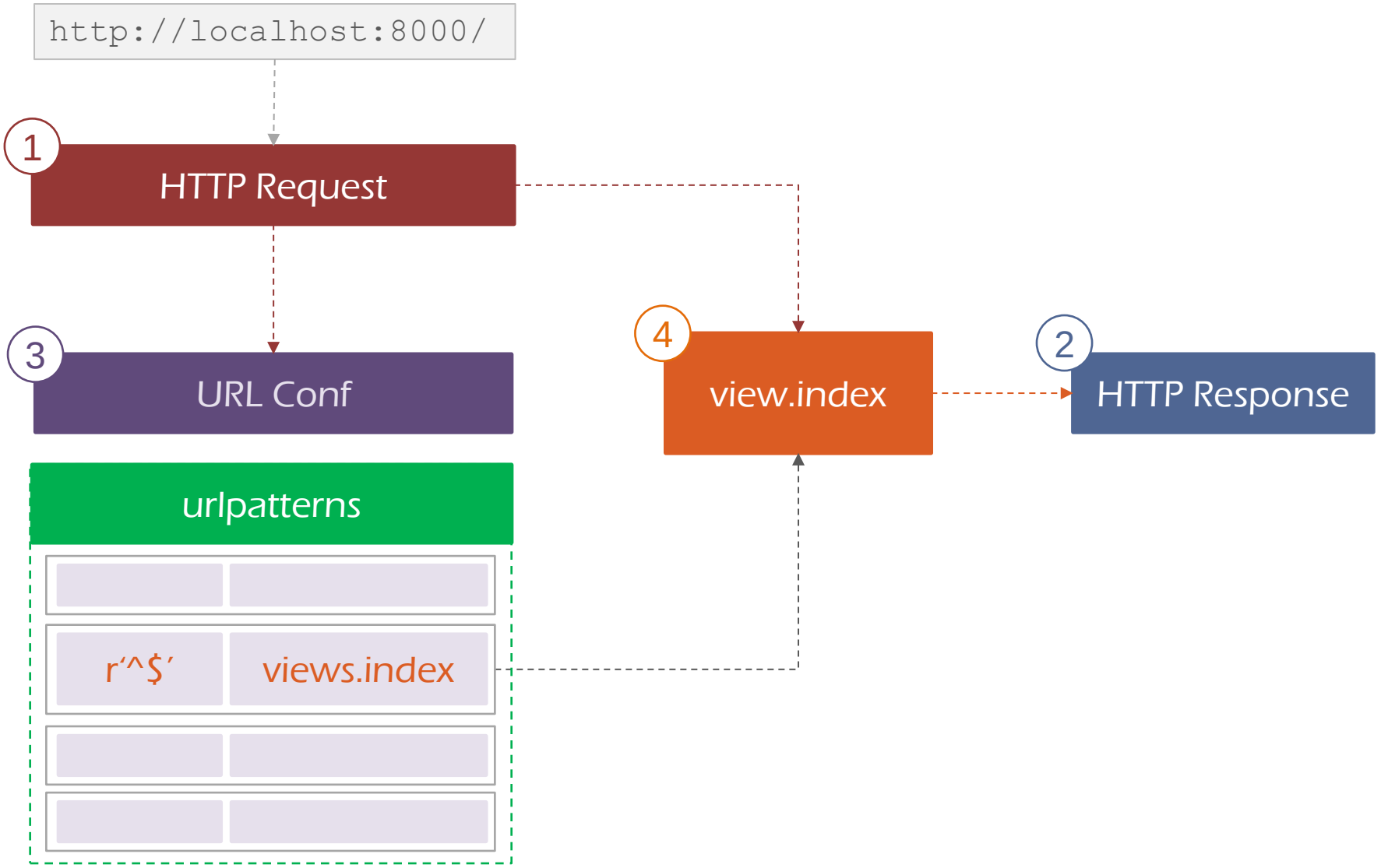
Perform Raw SQL

```
User.objects.raw('SELECT * FROM users_user')
```

Views

The Controller of Django

Intro


http://localhost:8000/
1
HTTP Request
3
URL Conf
urlpatterns
r'^$'
views.index
4
view.index
2
HTTP Response

Create a View

views.py

```
from django.http import HttpResponse

def index(request):

        return HttpResponse("Hello World")



def getUser(request, user_id):

        # Write some code to fetch user by it's id

...
```

Http Request

HttpRequest Object

`HttpRequest`

Attributes

`body`	An attribute that contain the request body
`method`	An attribute that contain the request method
`path`	An attribute that contain the request path
`GET`	An attribute that contain the GET request parameters
`POST`	An attribute that contain the POST request parameters
`COOKIES`	An attribute that contain the cookies
`FILES`	An attribute that contain the request File objects
`META`	An attribute that contain the request headers.

Methods

`get_host`	Return the Host name of the request
`is_ajax`	Return True if the request was made by XMLHttpRequest
`is_secure`	Return True if the request was made by https
`get_signed_cookie(key, salt='')`	Get the value of the signed(with salt) cookie

Http Response

Content

views.py

```
from django.http import HttpResponse

def index(request):

        res = HttpResponse("Hello World")

        res.write('<p> Hello Open Source </p>')

        return res
```

Hello World

Hello Open Source

Headers

views.py

```
from django.http import HttpResponse

def index(request):

        res = HttpResponse("Hello World")

        res.write('<p> Hello Open Source </p>')

        res['content-type'] = 'text/plain'

        return res
```

Hello World Hello Open Source

Cookies

views.py

```
from django.http import HttpResponse

def index(request):

        res = HttpResponse("Hello World")

        res.write('<p> Hello Open Source </p>')

        res['content-type'] = 'text/plain'

        res.set_cookie('name', 'Ahmed')

        return res
```

JsonResponse

views.py

```
from django.http import JsonResponse

def index(request):

        return JsonResponse({'name': 'Ahmed'})
```

HTTPResponseRedirect

views.py

```
from django.http import HttpResponseRedirect, HttpResponse

def index(request): #view for /

        return HttpResponseRedirect("/posts")


def get_posts(request): #view for /posts

        return HttpResponse ("Here are your posts")
```

Here are your posts

Urls

The way server understands client

Intro

mysite/settings.py

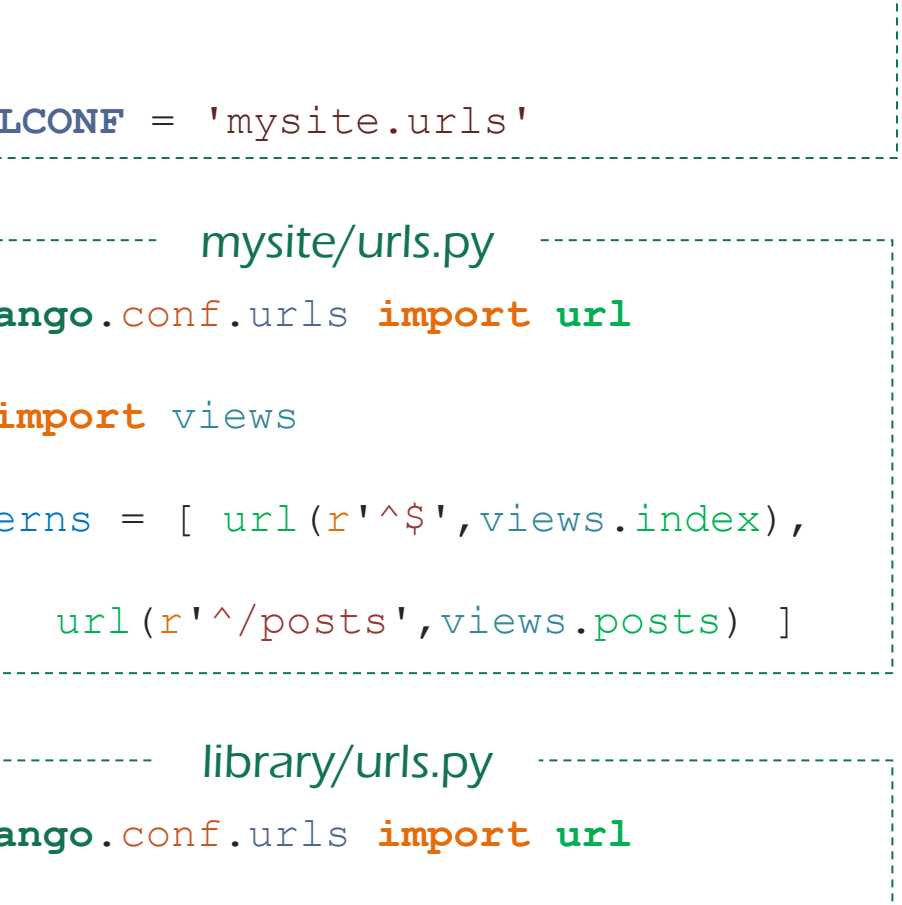
```
...

ROOT_URLCONF = 'mysite.urls'
```

mysite/urls.py

```
from django.conf.urls import url

from . import views

urlpatterns = [ url(r'^$',views.index),

          url(r'^/posts',views.posts) ]
```

library/urls.py

```
from django.conf.urls import url

from . import views

urlpatterns = [ url(r'^$',views.index)]
```



url

```
url(regex, view, kwargs=None, name=None)
```

`regex`	The regex pattern that identify the url.
`view`	The view function that handle that url pattern
`kwargs`	Some Extra options to be used in the view and the template
`name`	The URL pattern name.

Url *args & **kwargs

*args

http://localhost:8000/post/2/comment/5

```
url(r'^post/([0-9]+)/comment/([0-9]+)$', view.index)

# view.py
def index(request,*args):
        print('Post ID', args[0])       #2
        print('Comment ID', args[1])    #5
```

http://localhost:8000/post/4

*kwargs

```
url(r'^post/(?P<post_id>[0-9]+)$' , view.index)

#view.index
def index(request, post_id):
        print('Post ID', post_id)    #4
```

Include

```
include(module, namespace=None)

or

include(pattern_list)
```

`module`	*The module that you import the urlspatterns from.*
`namespace`	*The new custom namespace for your imported urlpatterns*
`Pattern_list`	*A hard coded urlpatterns list*

Include example

mysite/urls.py

```
from django.conf.urls import url, include

from . import views

urlpatterns = [

  url(r'^$',views.index),

  url(r'^/posts',views.posts),

  url(r'^/library', include('library.urls'))

]
```

library/urls.py

```
from django.conf.urls import url

from . import views

urlpatterns = [ url(r'^$',views.index)]
```

urlpatterns

r'^$'	views.index
r'^/posts'	views.posts
r'^/library'	library.urls

Shortcuts

It used for saving coding time

render

```
render(request, templateName, context=None, content_type=None)
```

views.py

```
from django.shortcuts import render

# view.py
def index(request):
    # Your View Body and Actions
    return render(request, 'persons/index.html', {'name': 'Ahmed'})
```

redirect

```
redirect(to, *args, **kargs)
```

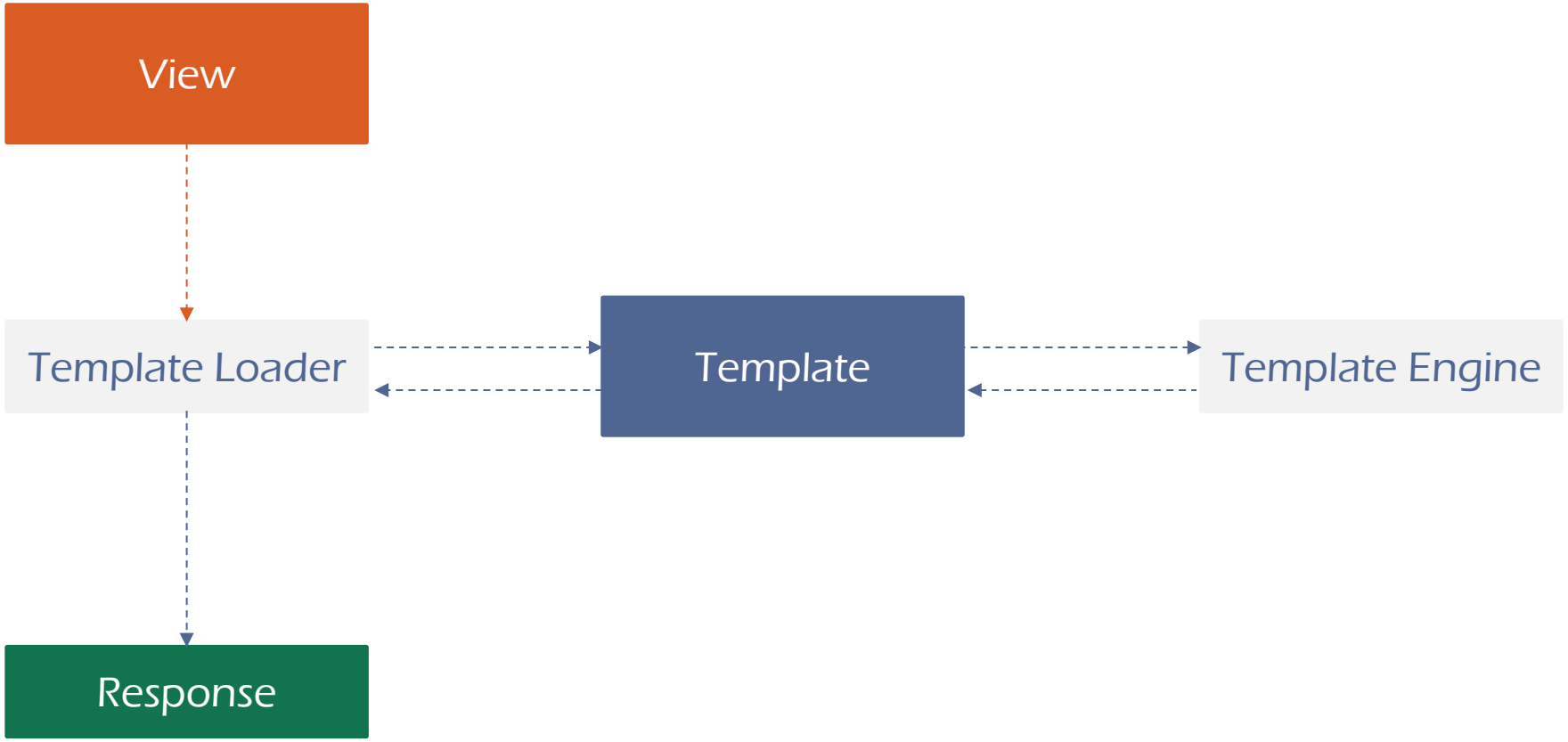
views.py

```
from django.shortcuts import redirect

# view.py
def index(request):
    # Your View Body and Actions
    return redirect('/persons', {'name': 'Ahmed'})
```

Templates

This is what client see

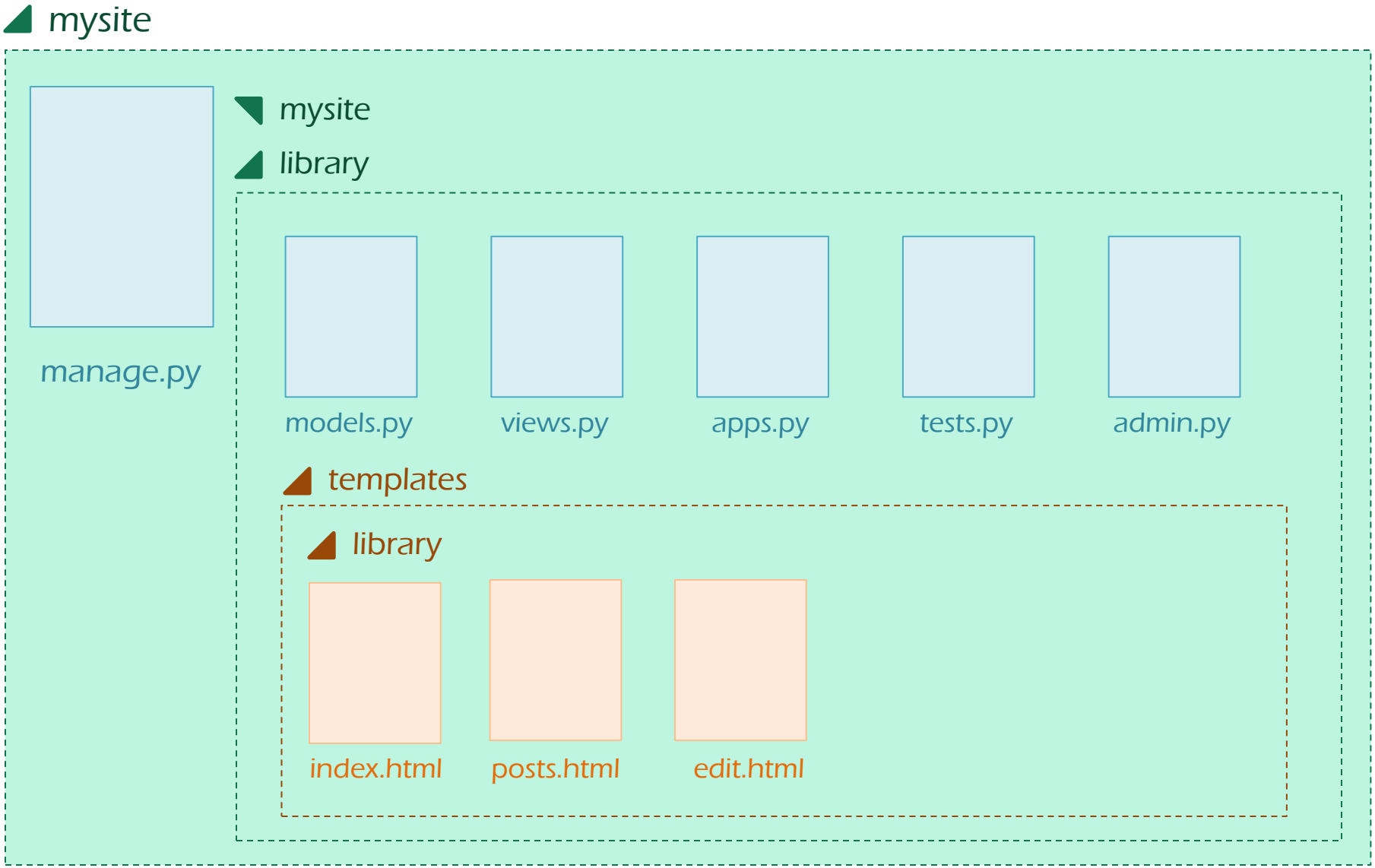
Intro



Example

```
<html>
  <head>
    <title>{% block title %}Default title{% endblock %}</title>
  </head>
  <body>
    <ul>
      {% for student in students %}
        {% block student %}
          <li> {{ student.name }} </li>
        {% endblock %}
      {% endfor %}
    </ul>
  </body>
</html>
```

Where To


mysite
mysite
library
manage.py
models.py
views.py
apps.py
tests.py
admin.py
templates
library
index.html
posts.html
edit.html

Settings

mysite/settings.py

```
TEMPLATES = [
    {
        #Define the Template Engine for the following templates
        'BACKEND':'django.template.backends.django.DjangoTemplates',
        #Define the directories that that the loader will search in
        'DIRS': [],
         #Define if the loader search in installed apps or not.
        'APP_DIRS': True,
    },
]
```

Static Files Management

Configuration

settings.py

```
INSTALLED_APPS = [
        # Other Apps,
        “django.contrib.staticfiles”
]
STATIC_URL = ‘/static/’
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
]
```

In your app create a directory called `static` and inside it put your app related static files

library/index.html

```
{% load static %}
<img src="{% static “library/example.jpg” %}" alt="My image"/>
```

Admin

Fully Implemented Admin Panel

Intro

Django Framework provide developers with a fully functional admin panel. That can easily be integrated with your project business

Create Super User


```
$ cd mysite
$ python manage.py createsuperuser
```

Register a model to Admin Panel

library/admin.py

```
from django.contrib import admin

from .models import Book

admin.site.register(Book)
```

Thank You