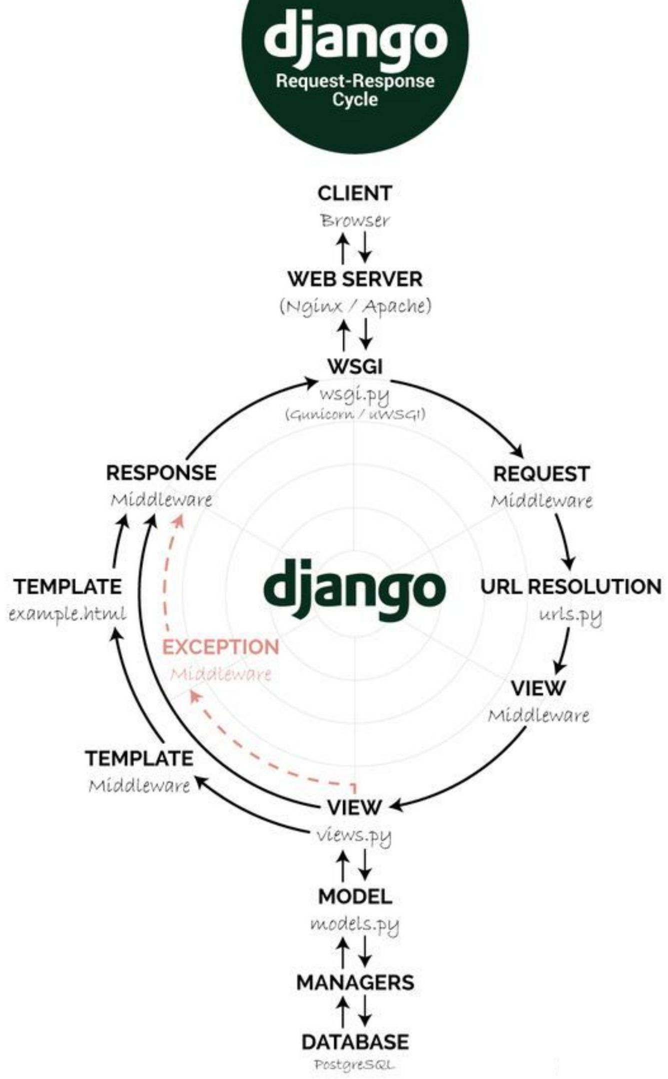
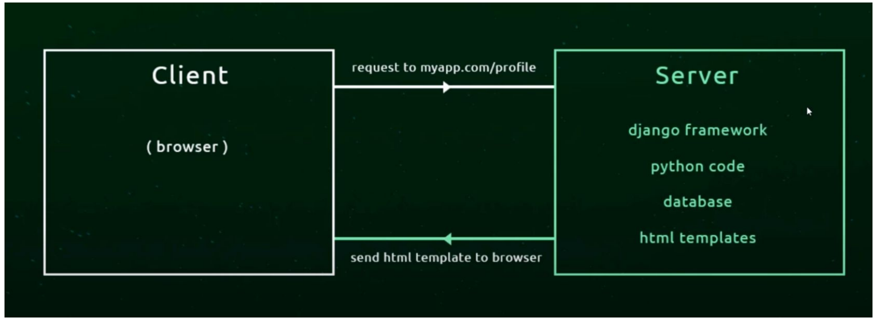
Django Models

Django Session-2

CLARUSWAY
WAY TO REINVENT YOURSELF



django
Request-Response Cycle
CLIENT
Browser
WEB SERVER
(Nginx / Apache)
WSGI
wsgi.py
(Gunicorn / uWSGI)
REQUEST
Middleware
URL RESOLUTION
urls.py
VIEW
Middleware
VIEW
views.py
MODEL
models.py
MANAGERS
DATABASE
PostgreSQL
TEMPLATE
Middleware
EXCEPTION
Middleware
TEMPLATE
example.html
RESPONSE
Middleware
django

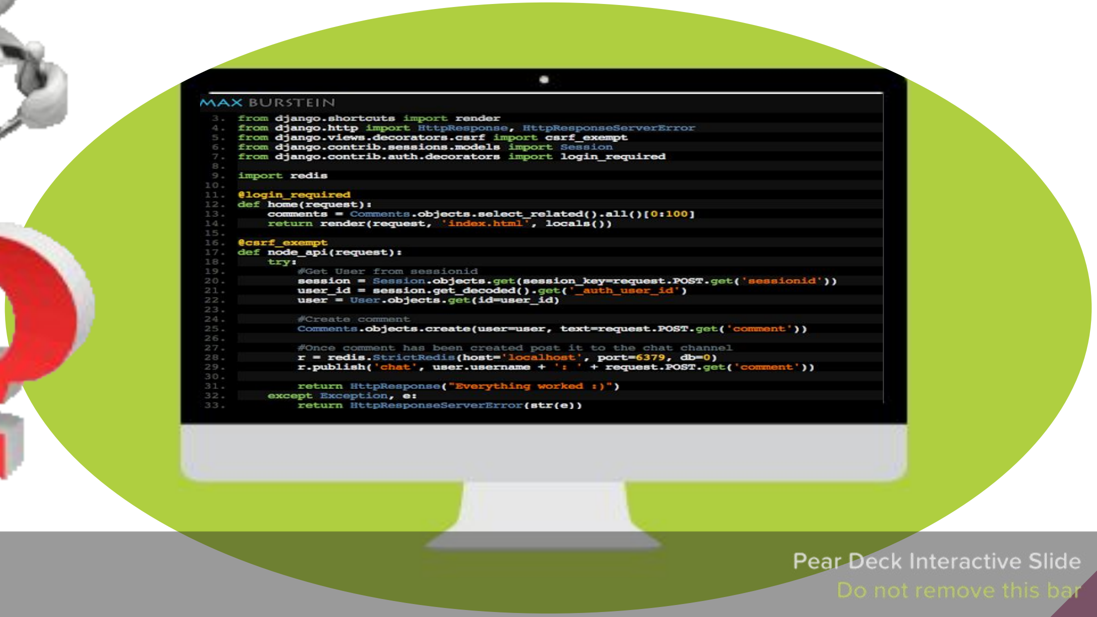


Client
(browser)
request to myapp.com/profile
send html template to browser
Server
django framework
python code
database
html templates

Table of Contents

Kahoot!

What is Model?

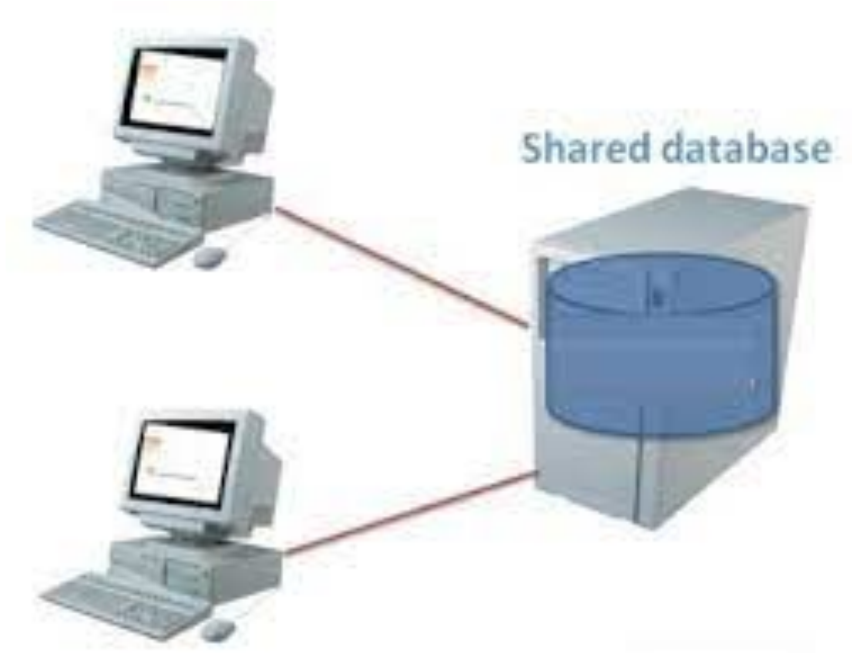


MAX BURSTEIN

```
from django.shortcuts import render
from django.http import HttpResponse, HttpResponseServerError
from django.views.decorators.csrf import csrf_exempt
from django.contrib.sessions.models import Session
from django.contrib.auth.decorators import login_required

import redis

@login_required
def home(request):
    comments = Comments.objects.select_related().all()[0:100]
    return render(request, 'index.html', locals())

@csrf_exempt
def node_api(request):
    try:
        #Get User from sessionid
        session = Session.objects.get(session_key=request.POST.get('sessionid'))
        user_id = session.get_decoded().get('_auth_user_id')
        user = User.objects.get(id=user_id)

        #Create comment
        Comments.objects.create(user=user, text=request.POST.get('comment'))

        #Once comment has been created post it to the chat channel
        r = redis.StrictRedis(host='localhost', port=6379, db=0)
        r.publish('chat', user.username + ': ' + request.POST.get('comment'))

        return HttpResponse("Everything worked :)")
    except Exception, e:
        return HttpResponseServerError(str(e))
```

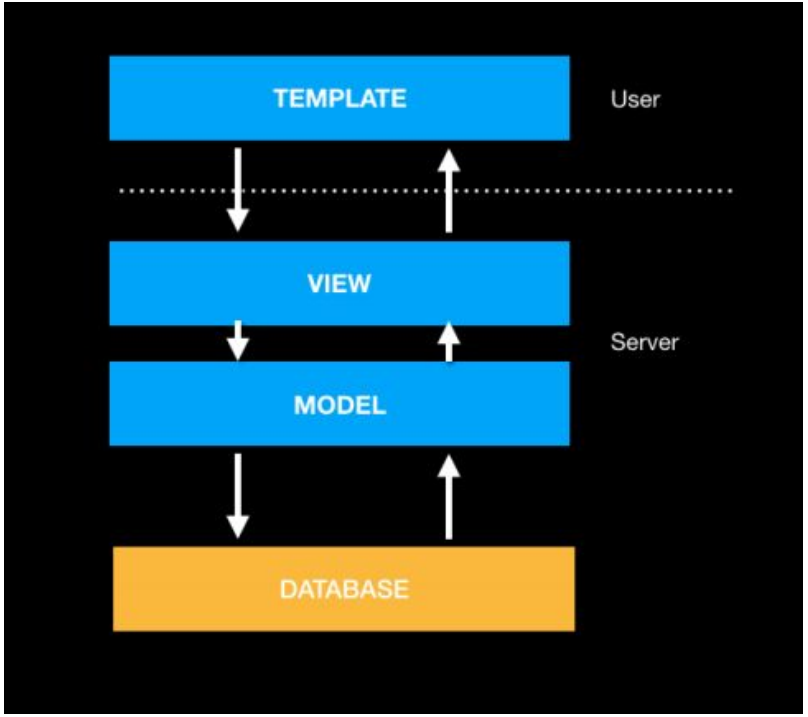
What is Model?

An essential part of any website is the ability to accept information from a user and input it into a database and retrieve information from a database and use it to generate content for the user.



CLARUSWAY
WAY TO REINVENT YOURSELF

What is Model?

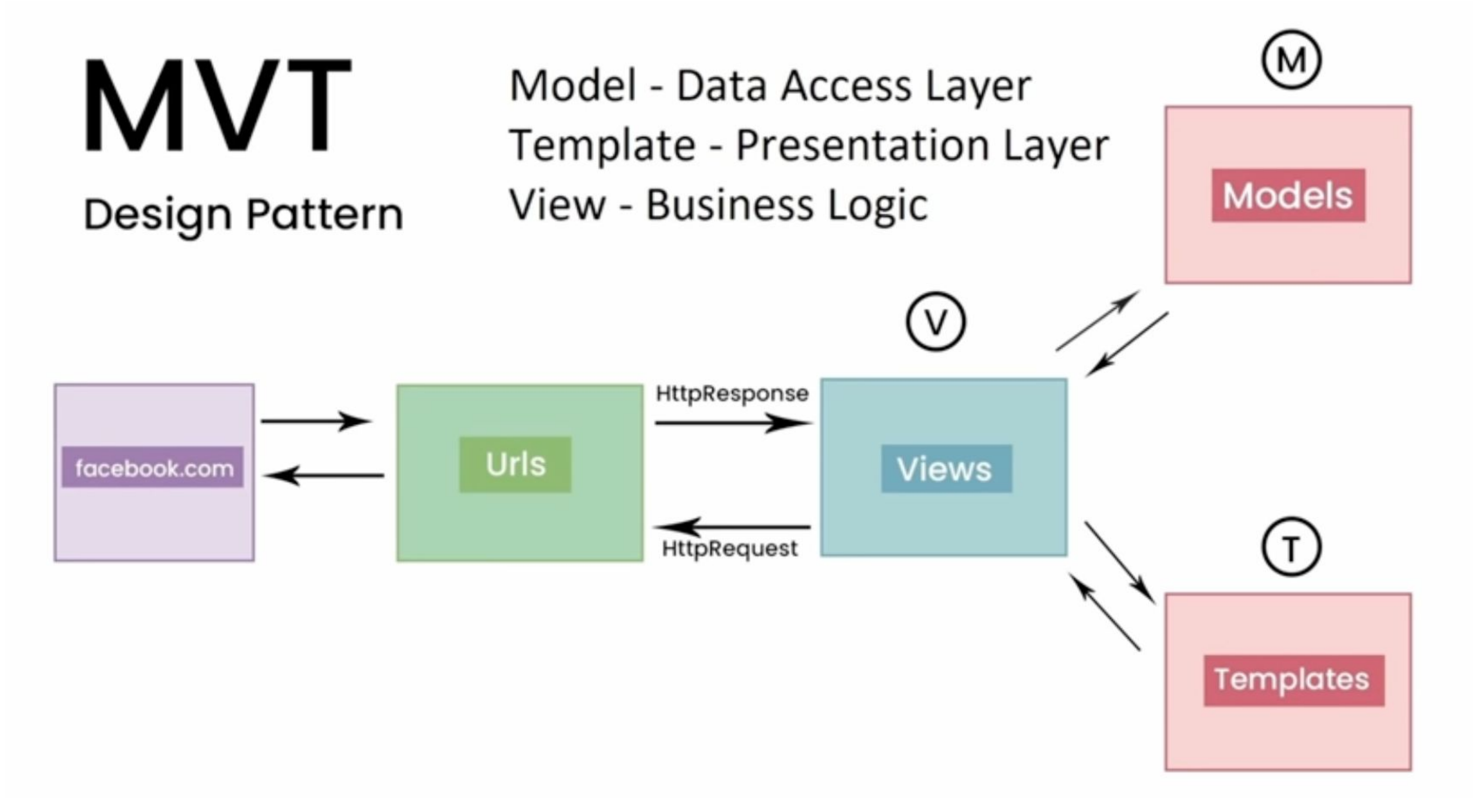


MVT stands for Model View Template
The MVT (Model View Template) is a software design pattern in Django’s architecture that consists of three components:

- Model: responsible for the database (ORM)
- Template: responsible for the presentation layer & user interface
- View: responsible for the business logic

CLARUSWAY
WAY TO REINVENT YOURSELF

What is Model?



What is Model?

- We use Models to incorporate a database into a Django Project.
- Django comes equipped with SQLite.
- SQLite will work for our simple examples, but Django can connect to a variety of SQL engine backends!

What is Model?

- In the settings.py file you can edit the ENGINE parameter used for DATABASES
- To create an actual model, we use a class structure inside of the relevant applications models.py file

What is Model?

- A model is the single, definitive source of information about your data.
- Each model is a Python class that subclasses django.db.models.Model
- We can use the Django admin panel to create, retrieve, update or delete model fields

CLARUSWAY
WAY TO REINVENT YOURSELF

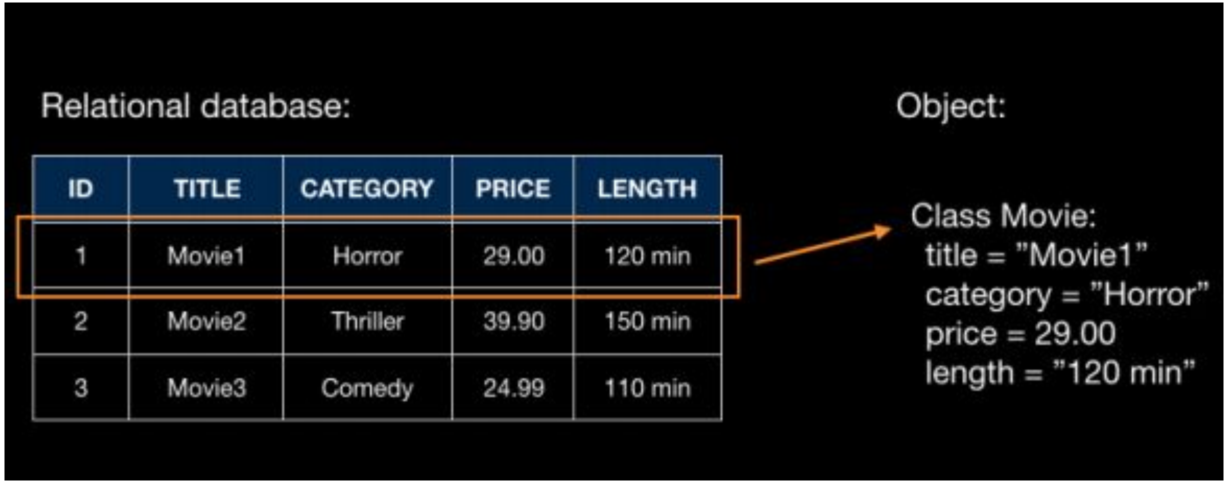
What is ORM?

- ORM stands for “**Object Relational Mapping**” and is a neat way to interact with the database
- Every class defined in the models file represents a database table
- Every defined property (field) represents a column in the table
- Each object is created based on a particular model, without having to write any SQL statement
- Advantages for this approach: simplified and cleaner solution, that is based on shorter and more readable querysets
- What is a Queryset? It is a list of objects of a particular model accessed from the database based on given criteria.

What is ORM?

Django ORM allows us to use django queries instead of using plain SQL statements, so i.e. if we want to access all the movies from the horror category we would use

SQL $\rightarrow$ SELECT * FROM Movies WHERE category=”Horror”
django queryset $\rightarrow$ Movie.objects.filter(category=”Horror”)



Relational database:

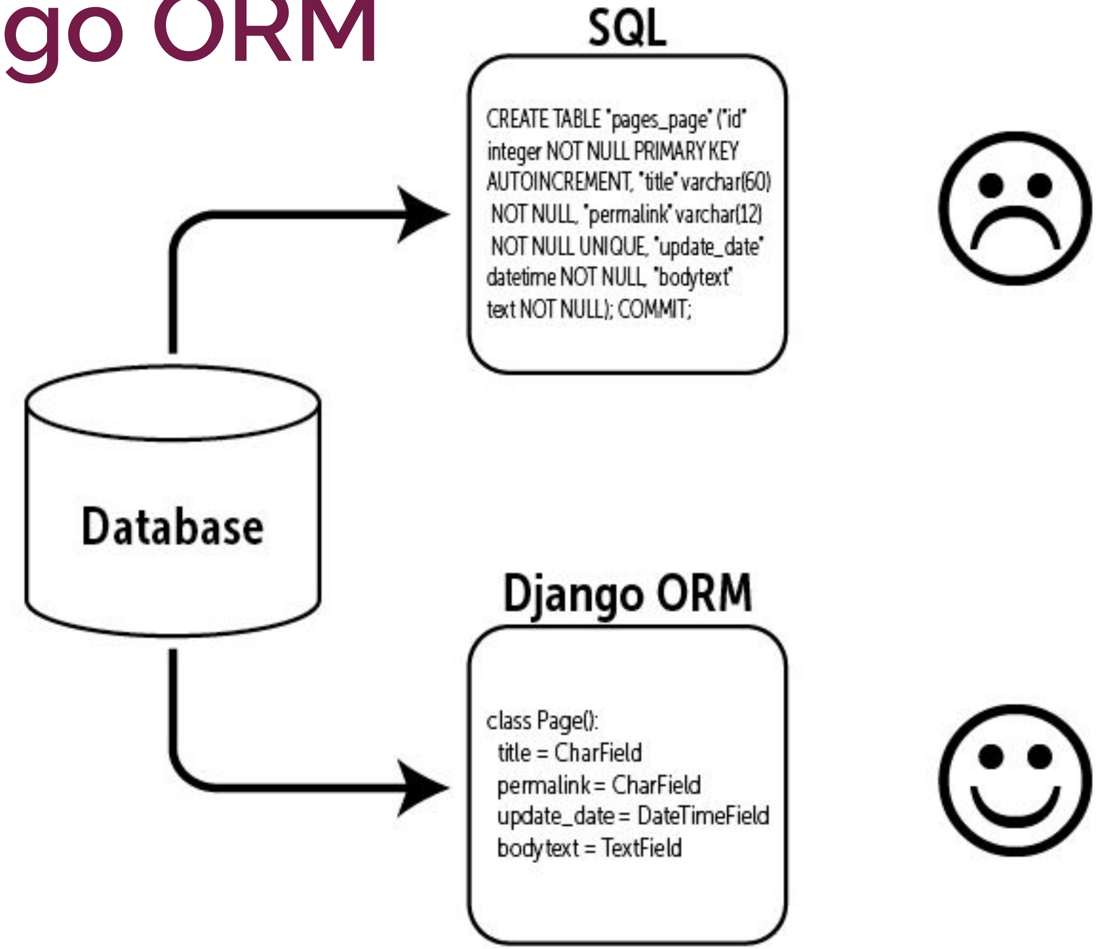
ID	TITLE	CATEGORY	PRICE	LENGTH
1	Movie1	Horror	29.00	120 min
2	Movie2	Thriller	39.90	150 min
3	Movie3	Comedy	24.99	110 min

Object:

```
Class Movie:
  title = "Movie1"
  category = "Horror"
  price = 29.00
  length = "120 min"
```

CLARUSWAY
WAY TO REINVENT YOURSELF

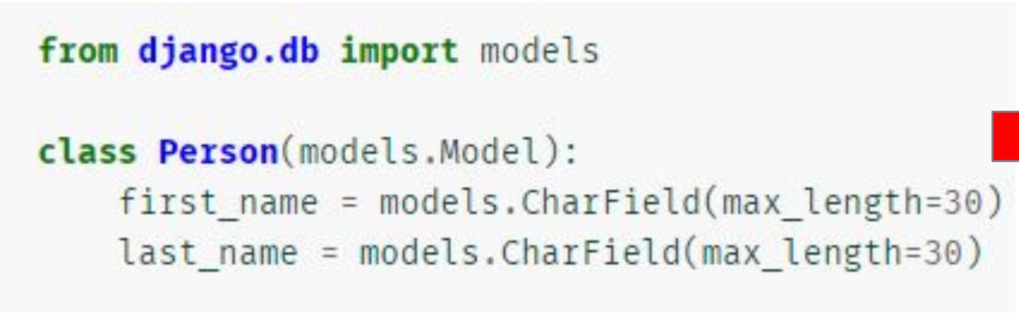
Django ORM



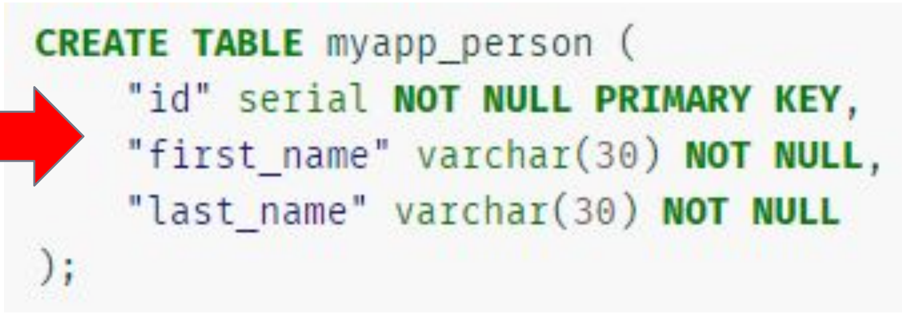
```
CREATE TABLE "pages_page" ("id"
integer NOT NULL PRIMARY KEY
AUTOINCREMENT, "title" varchar(60)
 NOT NULL, "permalink" varchar(12)
 NOT NULL UNIQUE, "update_date"
datetime NOT NULL, "bodytext"
text NOT NULL); COMMIT;
```

```
class Page():
  title = CharField
  permalink = CharField
  update_date = DateTimeField
  bodytext = TextField
```

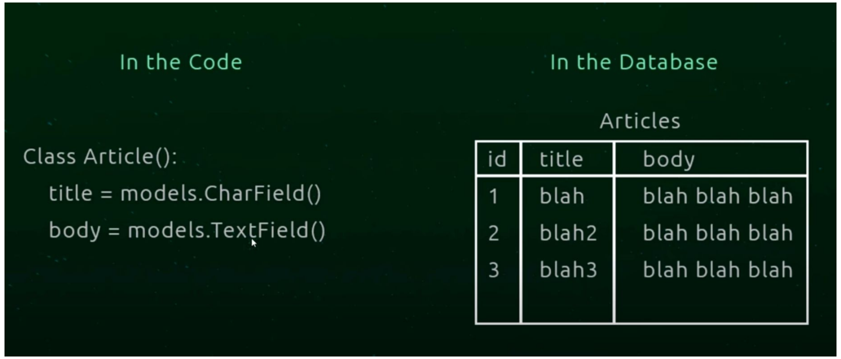
What is Model?



```
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```



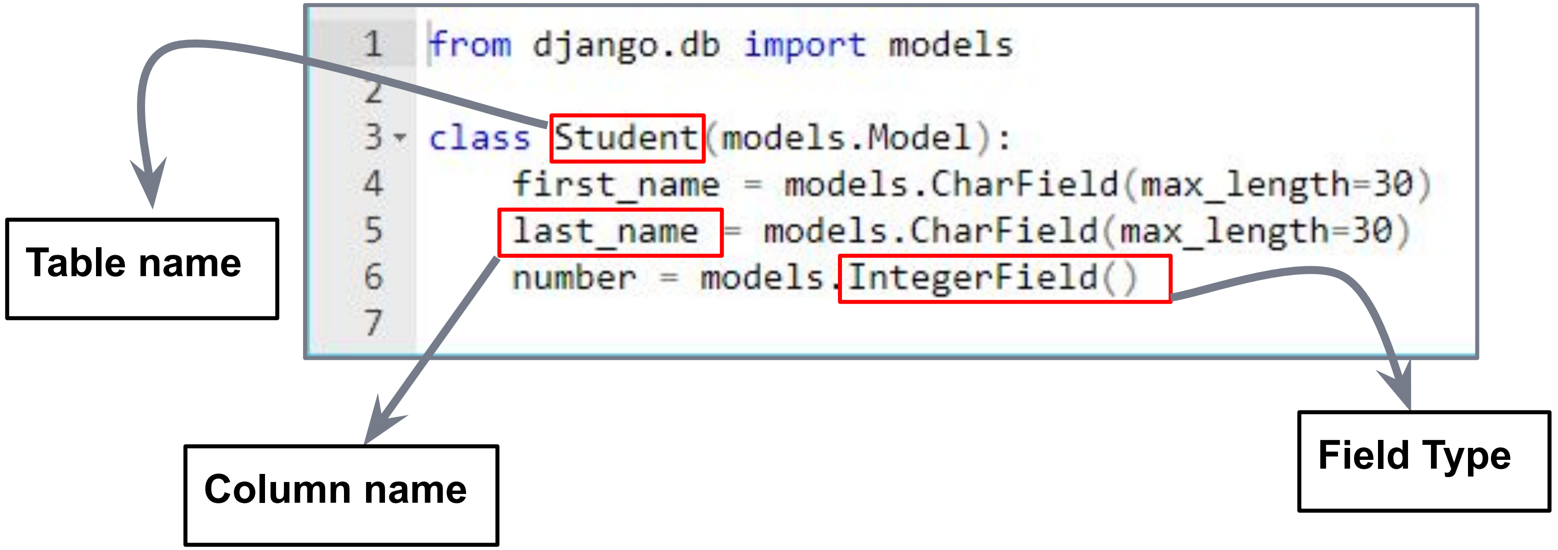
```
CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```



```
Class Article():
    title = models.CharField()
    body = models.TextField()
```

id	title	body
1	blah	blah blah blah
2	blah2	blah blah blah
3	blah3	blah blah blah

What is Model?



```python
from django.db import models

class Student(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    number = models.IntegerField()
```

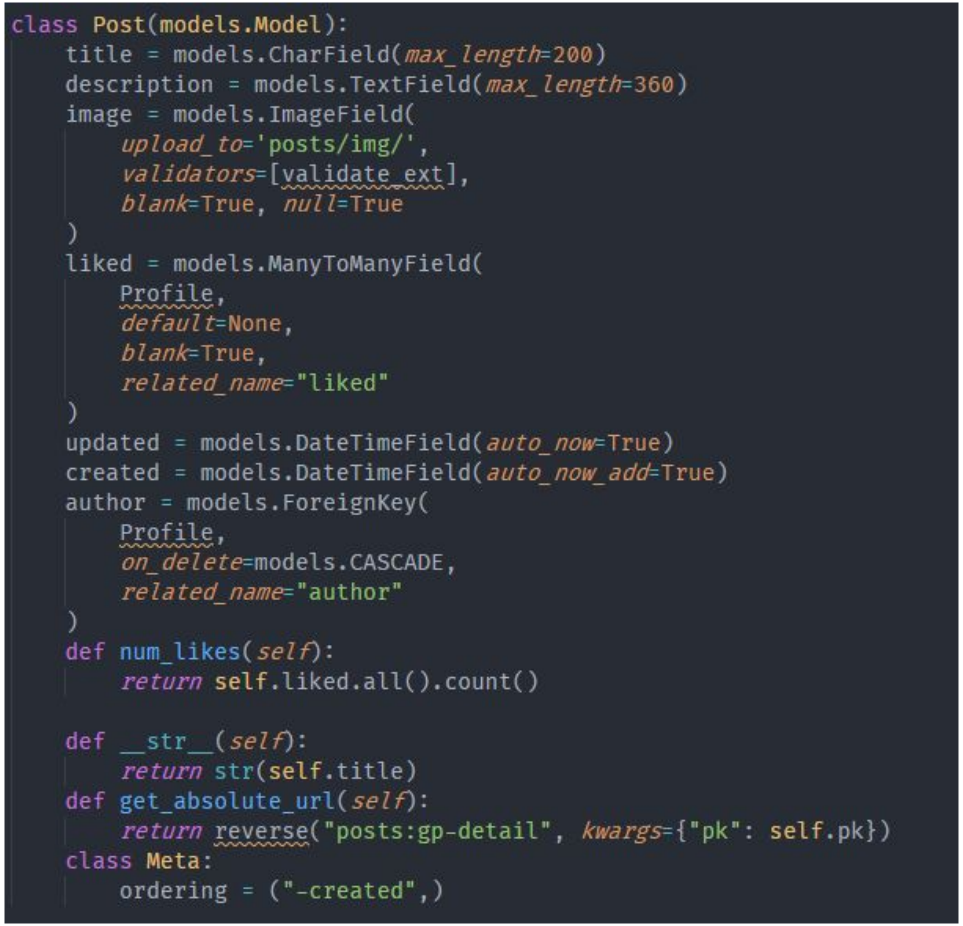
Field types

Field options

Relationships

CLARUSWAY
WAY TO REINVENT YOURSELF

What is Model?



```
class Post(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField(max_length=360)
    image = models.ImageField(
        upload_to='posts/img/',
        validators=[validate_ext],
        blank=True, null=True
    )
    liked = models.ManyToManyField(
        Profile,
        default=None,
        blank=True,
        related_name="liked"
    )
    updated = models.DateTimeField(auto_now=True)
    created = models.DateTimeField(auto_now_add=True)
    author = models.ForeignKey(
        Profile,
        on_delete=models.CASCADE,
        related_name="author"
    )
    def num_likes(self):
        return self.liked.all().count()

    def __str__(self):
        return str(self.title)
    def get_absolute_url(self):
        return reverse("posts:gp-detail", kwargs={"pk": self.pk})
    class Meta:
        ordering = ("-created",)
```

CLARUSWAY
WAY TO REINVENT YOURSELF

Basics Field Types

Field Name	Description
AutoField	It’s an InterField that automatically increments
BigAutoField	fit numbers from 1 to 9223372036854775807.
BigIntegerField	It is a 64-bit integer, much like an IntegerField except that it is guaranteed to fit numbers from -9223372036854775808 to 9223372036854775807.
BinaryField	A field to store raw binary data.
BooleanField	A true/false field. The default form widget for this field is a CheckboxInput.

Basics Field Types

Field Name	Description
CharField	A field to store text based values.
DateField	A date, represented in Python by a datetime.date instance
DecimalField	It is a fixed-precision decimal number, represented in Python by a Decimal instance.
FileField	It is a file-upload field.
FloatField	It is a floating-point number represented in Python by a float instance.

Basics Field Types

Field Name	Description
ImageField	It inherits all attributes and methods from FileField, but also validates that the uploaded object is a valid image.
TimeField	A time, represented in Python by a datetime.time instance.
TextField	A large text field. The default form widget for this field is a Textarea.
FileField	It is a file-upload field.
FloatField	It is a floating-point number represented in Python by a float instance.

Field Options

Field Name	Description
null	If True, Django will store empty values as NULL in the database. Default is False.
blank	If True, the field is allowed to be blank. Default is False.
db_column	The name of the database column to use for this field. If this isn’t given, Django will use the field’s name.
default	The default value for the field. This can be a value or a callable object. If callable it will be called every time a new object is created.

Field Options

Field Name	Description
help_text	Extra “help” text to be displayed with the form widget. It’s useful for documentation even if your field isn’t used on a form.
primary_key	If True, this field is the primary key for the model.
editable	If False, the field will not be displayed in the admin or any other ModelForm. They are also skipped during model validation. Default is True.
error_messages	The error_messages argument lets you override the default messages that the field will raise. Pass in a dictionary with keys matching the error messages you want to override.

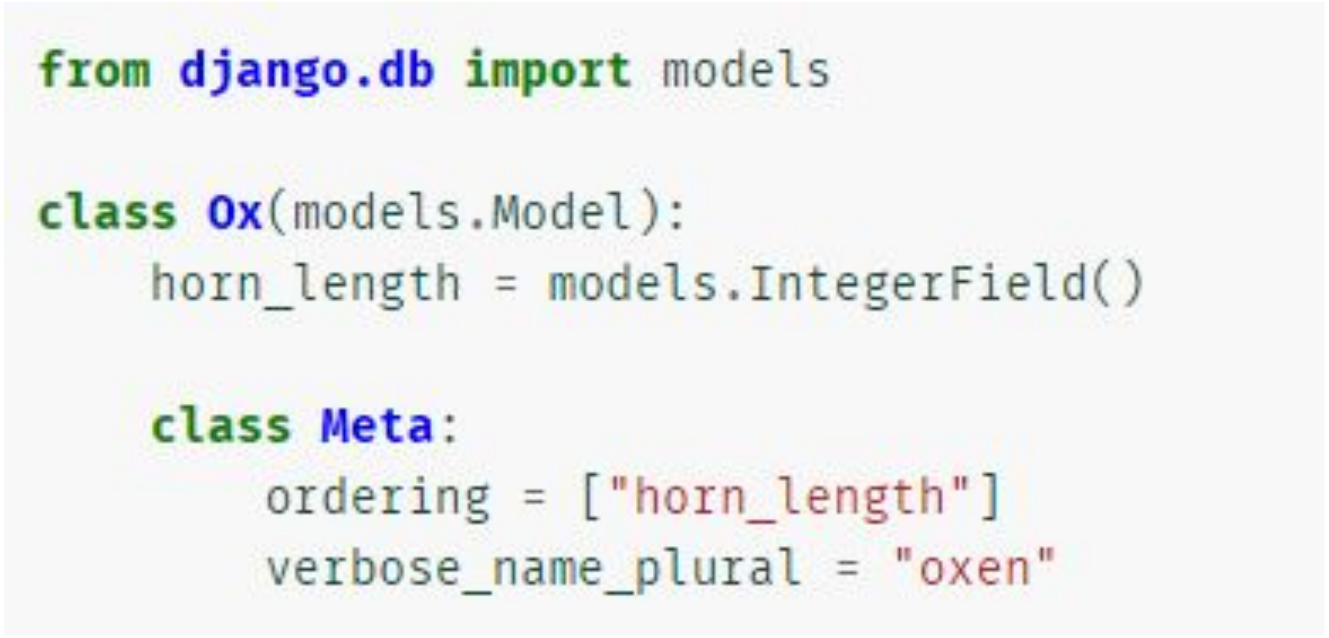
Field Options

Field Name	Description
verbose_name	A human-readable name for the field. If the verbose name isn’t given, Django will automatically create it using the field’s attribute name, converting underscores to spaces.
validators	A list of validators to run for this field. See the validators documentation for more information.
Unique	If True, this field must be unique throughout the table.

Relationship Fields

Field Name	Description
ForeignKey	A many-to-one relationship. Requires two positional arguments: the class to which the model is related and the on_delete option.
ManyToManyField	A many-to-many relationship. Requires a positional argument: the class to which the model is related, which works exactly the same as it does for ForeignKey, including recursive and lazy relationships.
OneToOneField	A one-to-one relationship. Conceptually, this is similar to a ForeignKey with unique=True, but the “reverse” side of the relation will directly return a single object.

Meta Options



```
from django.db import models

class Ox(models.Model):
    horn_length = models.IntegerField()

    class Meta:
        ordering = ["horn_length"]
        verbose_name_plural = "oxen"
```

Give your model metadata by using an inner **`class Meta`**,

adding class Meta to a model is completely optional.

CLARUSWAY
WAY TO REINVENT YOURSELF

What is Model?

```
python manage.py makemigrations
```

→ Tells Django you've made some changes to your models.

```
python manage.py migrate
```

→ Synchronizing the changes you made to your models with the schema in the database.

Interview Questions

What are models in Django?

What are models in Django?

A model in Django refers to a class that maps to a database table or database collection. Each attribute of the Django model class represents a database field. They are defined in app/models.py

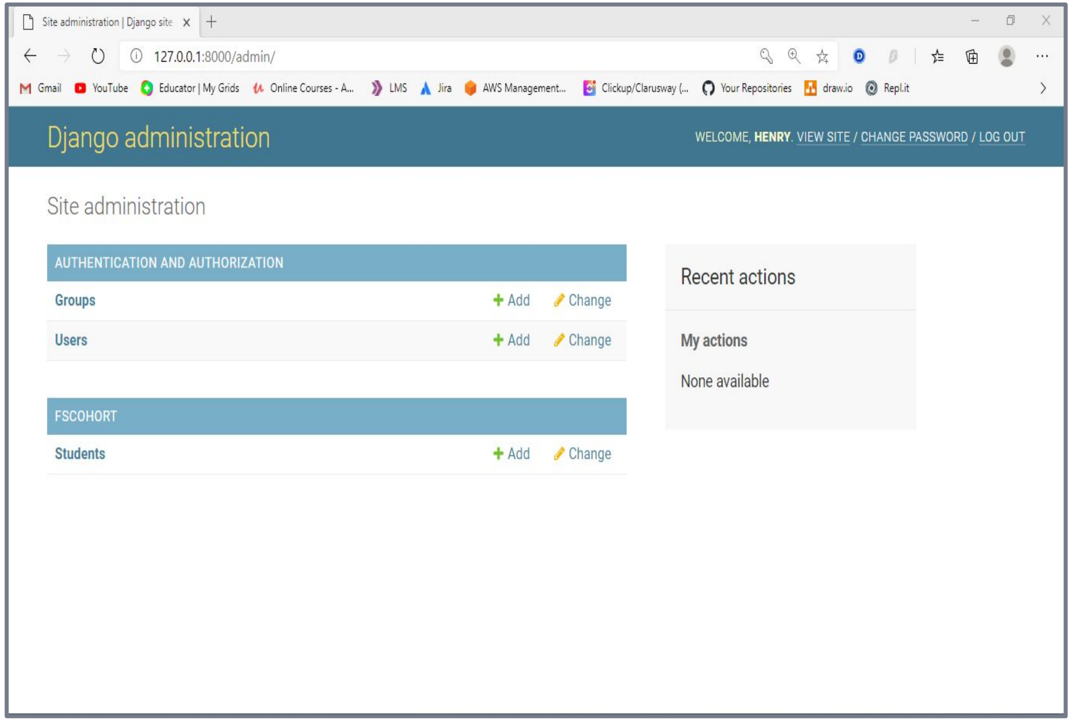
Every model inherits from django.db.models.Model

What is Model?

appname/admin.py

```
admin.site.register(Student)
```

```
python manage.py runserver
```



Django administration

WELCOME, **HENRY**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

FSCOHORT		
Students	+ Add	Change

Recent actions

My actions

None available

2

Databases

Databases

PostgreSQL

MySQL®

MariaDB

SQLite

ORACLE®
DATABASE

Databases

Django officially supports five databases:

SQLite is for early development and testing. Do not use in production.

- PostgreSQL
- MySQL
- SQLite
- Oracle
- MariaDB

There are also several third-party applications available if you need to connect to an unofficially supported database.

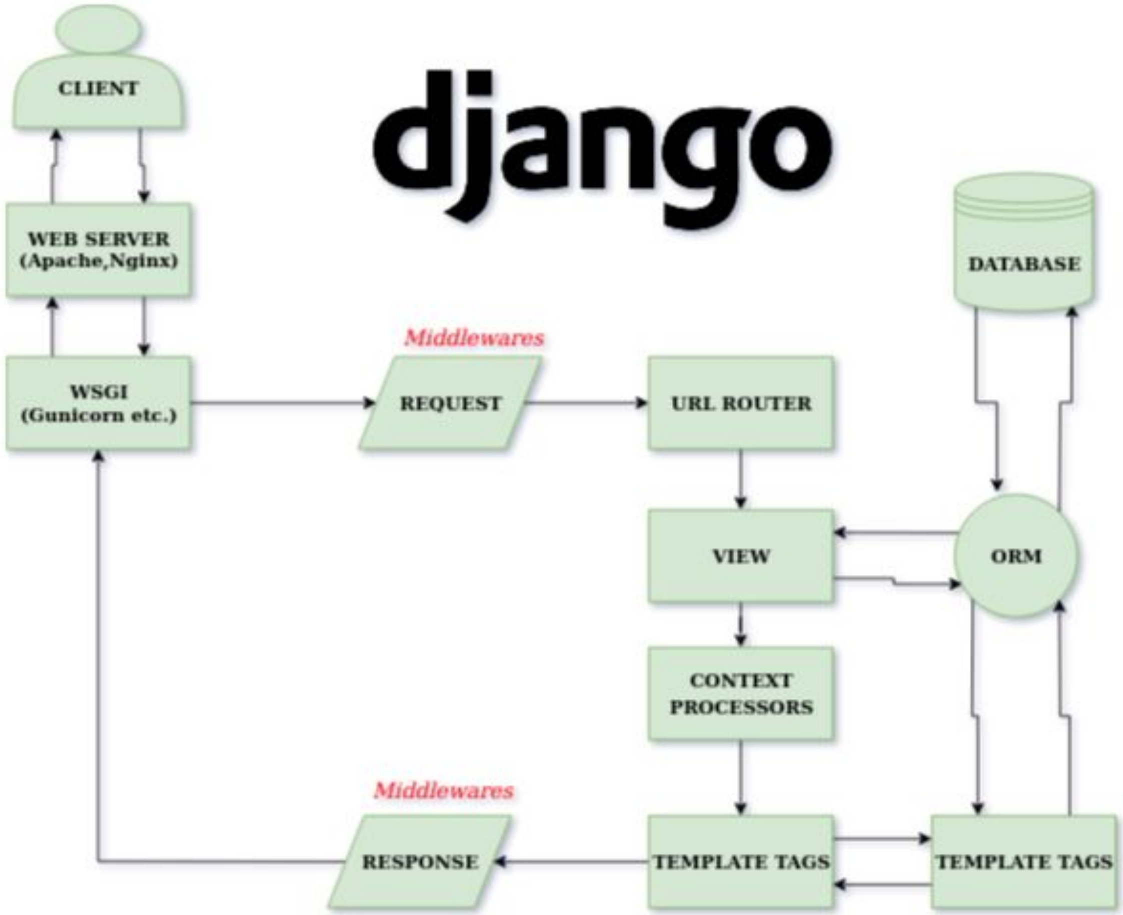
Databases

You can configure your database setting inside the settings.py file

```
DATABASES = {
    'default' : {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'database_name',
        'USER': 'username',
        'PASSWORD': 'password',
        'HOST': '127.0.0.1',
        'PORT': '3306',
        }
    }
}
```

THANKS!

Any questions?



Django Request-Response Cycle