Terminology

MVC *Model-View-Controller* is the “keep stuff separate” philosophy to separate out the code in a complicated web apps into 3 different categories

ORM *Object Relational Mapper* - the type of library that lets us create special classes (called *models*) that can be saved and retrieved from the database (e.g. SQLite). Writes SQL code, so we don’t have to.

migration Auto-generated code that uses the ORM to get the database sync’ed up with the latest additions to a project’s models.

applying migration Using a *migration* to get the DB up-to-date and ready for use.

MVT *Model-View-Template* are the three categories of code in a Django project

app A single Django-powered *project* can be split up into multiple *apps*. Each app can have a full vertical “slice” of models, views, and templates.

Models, Views, and Forms

models.py - Define a “Person” as having a name and email

```
class Person(models.Model):
    name = forms.CharField(max_length=64)
    email = forms.EmailField()
```

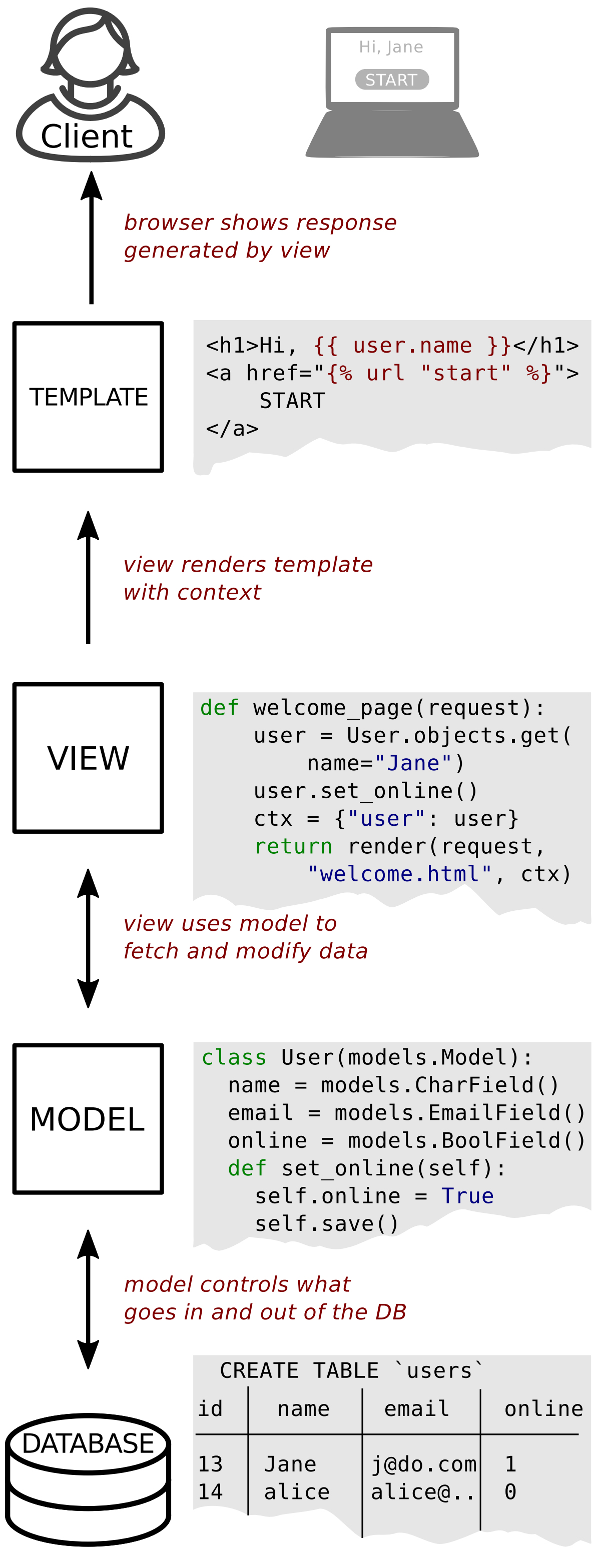
views.py - Example code for using a form and a model to gather and validate user input.

```
class NewPersonForm(forms.Form):
    name = forms.CharField(required=True)
    email = forms.EmailField()

# urls.py has: path("create/", views.person_create),
def person_create(request):
    if request.method == "GET":
        # Is initial GET: Create a blank form
        form = NewPersonForm()
    else:
        # Is POST: Create a form based on POST data
        form = NewPersonForm(request.POST)
        if form.is_valid():
            # If valid, create a new person & redirect
            person = Person()
            person.username = form.cleaned_data["name"]
            person.email = form.cleaned_data["email"]
            person.save()
            return redirect("/thanks/")
    ctx = {"form": form}
    return render(request, "create.html", ctx)
```

templates/create.html

```
<h1>Create new user</h1>
<form action="." method="post">
    {% csrf_token %}
    {{ form }}
    <button>Submit</button>
</form>
```


Client
Hi, Jane
START
browser shows response generated by view
TEMPLATE
view renders template with context
VIEW
view uses model to fetch and modify data
MODEL
model controls what goes in and out of the DB
DATABASE

```
<h1>Hi, {{ user.name }}</h1>
<a href="{% url "start" %}">
    START
</a>
```

```
def welcome_page(request):
    user = User.objects.get(
        name="Jane")
    user.set_online()
    ctx = {"user": user}
    return render(request,
        "welcome.html", ctx)
```

```
class User(models.Model):
  name = models.CharField()
  email = models.EmailField()
  online = models.BoolField()
  def set_online(self):
    self.online = True
    self.save()
```

CREATE TABLE `users`

id	name	email	online
13	Jane	j@do.com	1
14	alice	alice@..	0

model is the gate-keeper to data stored in the *database*

view defines *business logic* of your web app

template is the appearance of your site in HTML