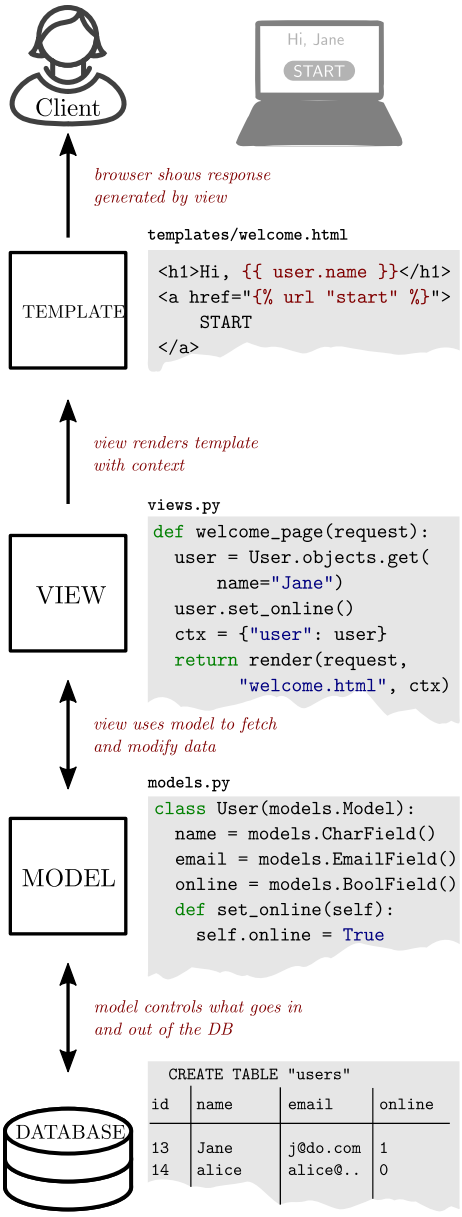

`urls.py`: Routing

```
from django.urls import path, include
from myapp import views
import accounts
urlpatterns = [
  path("/", views.welcome_page),
  path("/u/<int:uid>/",views.upage),
  include("acct/", accounts.urls),
]
```

`forms.py`: Form validation

```
from django import forms
class NewPersonForm(forms.Form):
  name = forms.EmailField()
```

`python manage.py` work-flow

```
startapp # Scaffold django "app"
runserver # Run test server
shell # Enter Python shell
dbshell # Enter Database shell
showmigrations # Migration status
makemigrations # Gen model changes
migrate # Apply migrations to DB
```

`models.py`: DB Schema

```
class Author(models.Model):
  # A simple text field
  name = models.CharField(
    max_length=100)
  # Define __str__ on your models
  def __str__(self): # and return
    return self.name # its title

class Book(models.Model):
  # Use a "ForeignKey" for a
  # ManyToOne relationship
  author = models.ForeignKey(
    Author, on_delete="CASCADE")
  # Store when created and modified
  created = models.DateTimeField(
    auto_now_add=True)
  updated = models.DateTimeField(
    auto_now=True)
  # TextField is for long text.
  # null=True, blank=True allows
  # values of "" and None
  blurb = models.TextField(
    null=True, blank=True)
  # Multiple-choice fields in pattern:
  # "internal code, external label"
  CATEGORIES = [
    ("fict", "Fiction"),
    ("nonfict", "Non-fiction")]
  category = models.CharField(
    max_length=10,
    default="fict",
    choices=CATEGORIES,
  )
  # Validators add custom checks
  num_stars = models.IntegerField(
    validators=[MaxValueValidator(5),
                MinValueValidator(1)])

  # ManyToMany relationships
class ReadingList(models.Model):
  books = models.ManyToManyField(Book)
```

`views.py`: Business logic

```
def welcome_page(request):
  return render(request,
                "hi.html", {})
def upage(request, uid):
  usr = Users.objects.get(id=uid)
  return render(request, "u.html", {
    "user": usr })
```

ORM: CRUD Examples

```
### CREATE: Save new book to DB
book = Book.objects.create(
  title="Oliver Twist",
  num_stars=4)
### READ: Get all fiction books
fiction_books = Book.objects\
  .filter(category="fict")
# Get all 4+ star books, newest first
new_good_books = Book.objects\
  .filter(num_stars__gt=3)\
  .order_by("-date")
### UPDATE: Change existing book(s)
book = Book.objects.get(title="1984")
book.num_stars = 5 # Updates a property
book.save() # Saves change to the DB
nonfict = Book.objects.filter(category="
nonfict.update(num_stars=5)  # Updates a
### DELETE: Delete one or more book(s)
book = Book.objects.get(title="1984")
book.delete() # Delete a single book
Book.objects.filter( # Delete multiple
   num_stars__lt=3).delete()
```

Templates

variables Use "." to access `dict` keys *and* properties *and* methods

```
<h2>Hi {{ user.username }}!</h2>
```

if

```
{% if age > 17 %}
    <p>You may continue.</p>
{% else %}
    <p>Too young.</p>
{% endif %}
```

for

```
{% for post in blog_posts %}
    <h2>{{ post.title }}</h2>
{% empty %}
    <em>No posts found</em>
{% endfor %}
```

extends & blocks Allows template variations: replace "block" placeholder in a `base.html`

```
{% extends "base.html" %}
{% block main_content %}
    <p>User: {{ user.name }}<p>
{% endblock main_content %}
```

filters `<p>Hi {{ name|upper }}</p>`

include `{% include "snippet.html" %}`

using forms

```
<form action="." method="POST">
  {% csrf_token %} {{ form }}
  <button>Save</button> </form>
```