DJANGO
UNCHAINED

EPISODE 2: DJANGO'S ARCHITECTURE

THE WEB FRAMEWORK FOR PERFECTIONISTS WITH DEADLINES.

NAHIM NASSER SHAH WARRAICH

WHO'S USING IT?

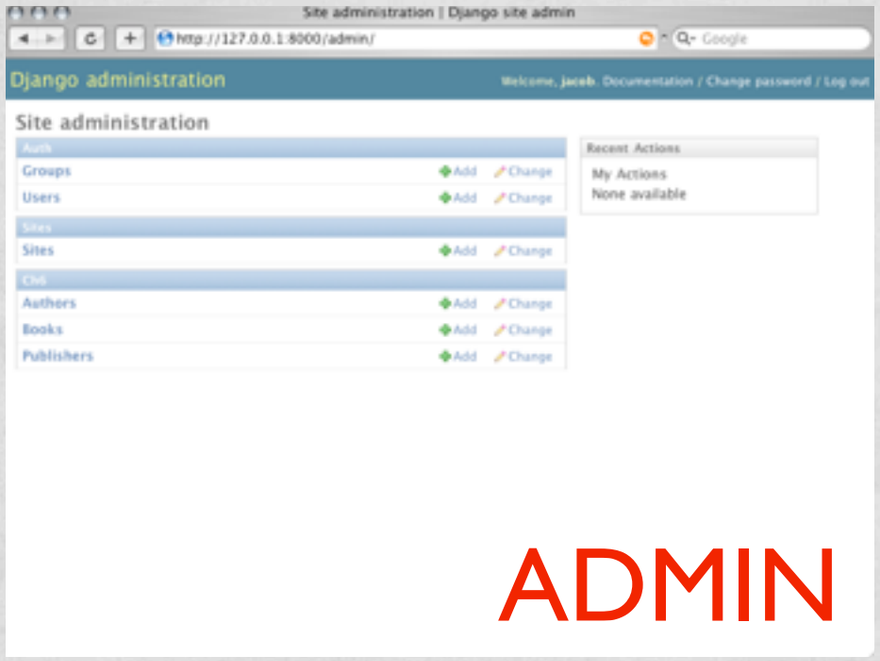
Instagram

Pinterest

DISQUS

“IF THE ROMANS HAD DJANGO,

THEY WOULD HAVE BUILT ROME IN A **DAY**”



ADMIN

```
{% block content %}

    {{ category.name }}

    <h1>Articles</h1>

{% for article in articles %}
    <div class="article">
        <h2>{{ article.title }}</h2>

        <div class="content">
            {{ article.content|markdown }}
            -written by {{ author }}
        </div>
    </div>
{% endfor %}
{% endblock %}
```

TEMPLATES

```
from django import forms
from django.db import models

class Game(models.Model):
    '''
        A simple Django model representing a Ping-P
        between two players. Games are sorted by da
    '''
    player_one = models.CharField('Player One', max
    player_two = models.CharField('Player Two', max
    player_one_score = models.IntegerField("Player
    player_two_score = models.IntegerField("Player
    played_on = models.DateTimeField('Date/Time Pla

    class Meta:
        ordering = ['-played_on']

    def winner(self):
        if self.player_one_score > self.player_two
```

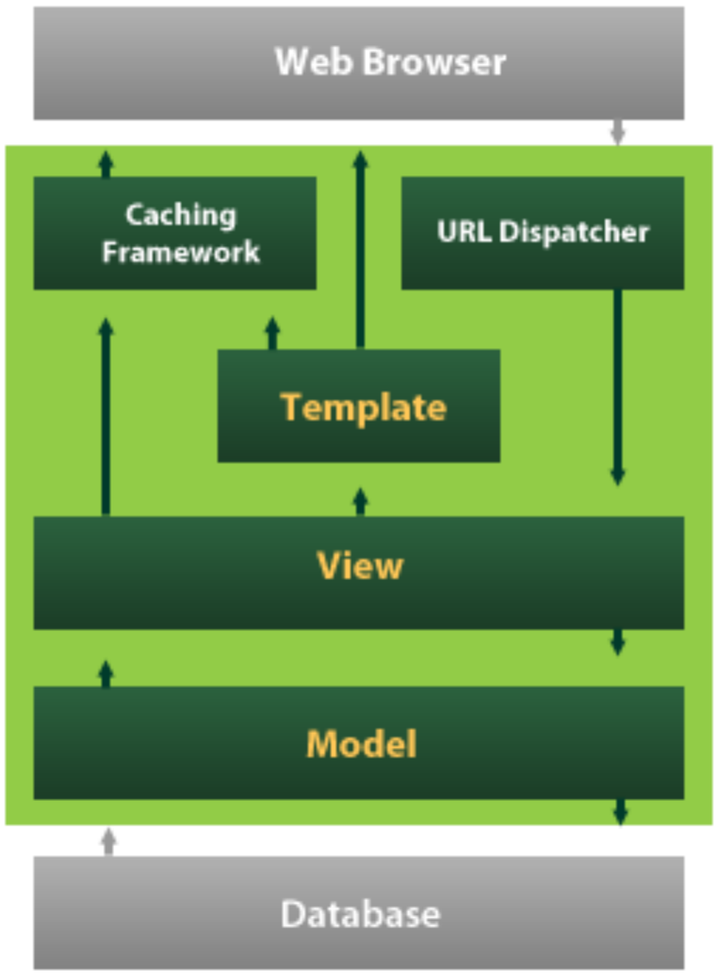
ORM

TOP FEATURES

STABILITY
SCALABILITY
DATA INTEGRITY

DIVING INTO THE ARCHITECTURE

Model-View-Template (MVT) Architecture



1. The URL dispatcher (urls.py) maps the requested URL to a view function and calls it. If caching is enabled, the view function can check to see if a cached version of the page exists and bypass all further steps, returning the cached version, instead. Note that this page-level caching is only one available caching option in Django. You can cache more granularly, as well.

2. The view function (usually in views.py) performs the requested action, which typically involves reading or writing to the database. It may include other tasks, as well.

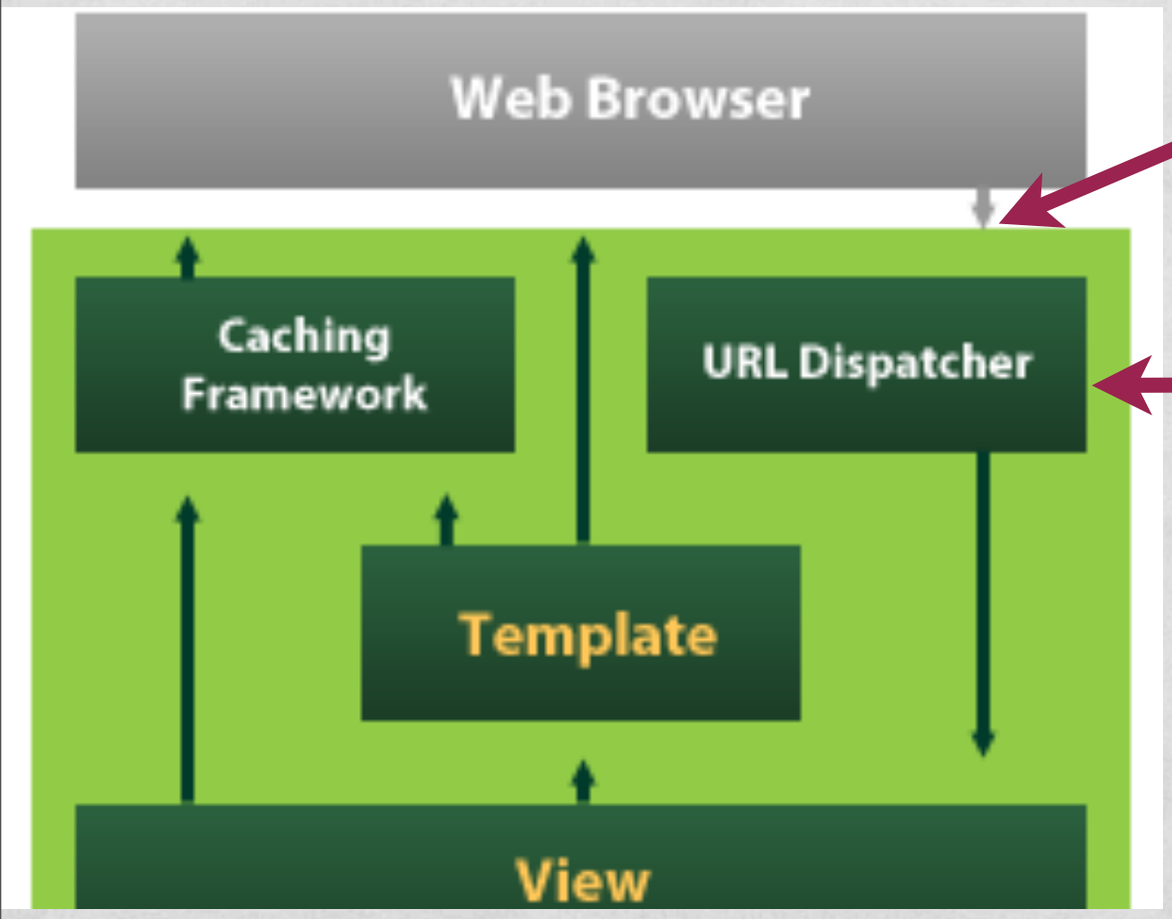
3. The model (usually in models.py) defines the data in Python and interacts with it. Although typically contained in a relational database (MySQL, PostgreSQL, SQLite, etc.), other data storage mechanisims are possible as well (XML, text files, LDAP, etc.).

4. After performing any requested tasks, the view returns an HTTP response object (usually after passing the data through a template) to the web browser. Optionally, the view can save a version of the HTTP response object in the caching system for a specified length of time.

5. Templates typically return HTML pages. The Django template language offers HTML authors a simple-to-learn syntax while providing all the power needed for presentation logic.

Image from: http://packages.python.org/MyTARDIS/_images/DjangoArchitecture-JeffCroft.png

THE URL DISPATCHER



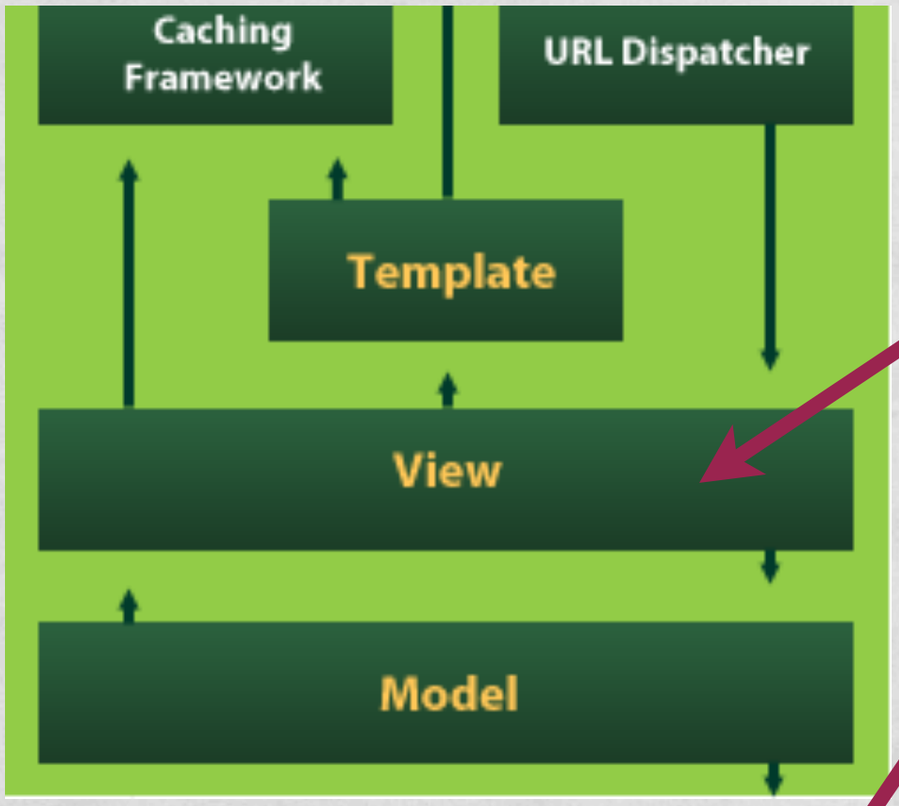
1. HTTP REQUEST MADE

2. URLS.PY

```
from django.conf.urls import patterns, include, url
from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    (r'^admin/', include(admin.site.urls)),
    (r'^login_test', 'userprofiles.views.test_login'),
    (r'^english_word/', 'gameplay.views.is_english_word'),
    (r'^test_authenticated', 'userprofiles.views.test_authenticated'),
    (r'^me', 'userprofiles.views.me'),
    (r'^users/new', 'userprofiles.views.register'),
    (r'^users/facebook', 'userprofiles.views.facebook'),
    (r'^users/friends', 'userprofiles.views.my_friends'),
    (r'^users/login', 'userprofiles.views.log_in'),
    (r'^users/logout', 'userprofiles.views.log_out'),
    (r'^sessionid', 'userprofiles.views.get_session_id'),
    (r'^games/submit-score', 'gameplay.views.submit_score'),
    (r'^games/og/', 'gameplay.views.game_og'),
    (r'^games/new', 'gameplay.views.new_game'),
    (r'^games/end', 'gameplay.views.end_game'),
    (r'^games/time', 'gameplay.views.add_time'),
    (r'^games/guess', 'gameplay.views.make_a_guess'),
    (r'^games/mine', 'gameplay.views.my_games'),
    (r'^games/letters', 'gameplay.views.generate_letters'),
    (r'^games/(\d+)', 'gameplay.views.view_game'),
    (r'^validate_word', 'gameplay.views.validate_word'),
    (r'^', 'userprofiles.views.welcome'),
)
```

VIEW CALLBACK

DJANGO VIEWS



4. VIEWS.PY

```
25  @login_required
26  def end_game(request,*args,**kwargs):
27      """
28      A POST request that should provide the necessary paramete
29
30      game_id - game id that is to be ended
31      """
32      user = request.user
33      if request.method == 'POST':
34          game_id = int(request.REQUEST.get('game_id',-1))
35          response = {'success':False,'message':''}
36          """
37          Validate parameters, make sure the user can end the g
38          """
39          try:
40              game = Game.objects.get(id=game_id)
41          except Game.DoesNotExist:
42              response['message'] = 'Game with id %s not found'
43              return render_json(response,'NotFound')
44
45          if(game.ended):
46              response['message'] = 'Game has ended already.'
47              return render_json(response,'BadRequest')
48
49          if game.player2 != user:
50              response['message'] = 'User is not guessing this
51              return render_json(response,'BadRequest')
52
53          try:
54              game.ended = timezone.now()
55              game.save()
56          except Exception,e:
57              print e
58          response['success']=True
59          response['message']= 'Successfully ended game'
60          return render_json(response)
61      else:
```

Notice the function signature & parameters

DJANGO MODELS

```python
@login_required
def end_game(request,*args,**kwargs):
    """
    A POST request that should provide the necessary parameters for the end game.

    game_id - game id that is to be ended
    """
    user = request.user
    if request.method == 'POST':
        game_id = int(request.REQUEST.get('game_id',-1))
        response = {'success':False,'message':''}
        """
        Validate parameters, make sure the user can end the gameid provided
        """
        try:
            game = Game.objects.get(id=game_id)
        except Game.DoesNotExist:
            response['message'] = 'Game with id %s not found'%game_id
            return render_json(response,'NotFound')

        if(game.ended):
            response['message'] = 'Game has ended already.'
            return render_json(response,'BadRequest')

        if game.player2 != user:
            response['message'] = 'User is not guessing this game.'
            return render_json(response,'BadRequest')

        try:
            game.ended = timezone.now()
            game.save()
        except Exception,e:
            print e
        response['success']=True
        response['message']= 'Successfully ended game'
        return render_json(response)
    else:
        return invalid_method(['POST'])
```

THIS IS A MODEL REFERENCE

Model manager

This queries the DB for a game object where id=game_id

DJANGO MODEL DEFINITIONS

MODELS.PY

```python
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User
import tools
import settings
from django.db.models import Q
from userprofiles.models import *

"""
A Game is between two users, with a word to guess (chosen by player 1
and a set of letters to choose from (which player 2 will guess).
Each game will consist of rounds which is player 2 guessing a word.
Each round will affect both user's scores (probably player 1's will g
"""

class Game(models.Model):

    player1 = models.ForeignKey(User, related_name="player1_game_set"
    player2 = models.ForeignKey(User, related_name="player2_game_set"
    letters = models.CharField(max_length=24)
    word = models.CharField(max_length=15)
    player1score = models.IntegerField(blank=True, null=True)
    player2score = models.IntegerField(blank=True, null=True)
    started = models.DateTimeField('time started')
    ended = models.DateTimeField('time ended',blank=True,null=True)
    solved = models.BooleanField(default=False)
    previous_game = models.ForeignKey('self',blank=True,null=True)
    time = models.IntegerField(default=0) #time used by the guessor
```

Relationships to other models can be defined

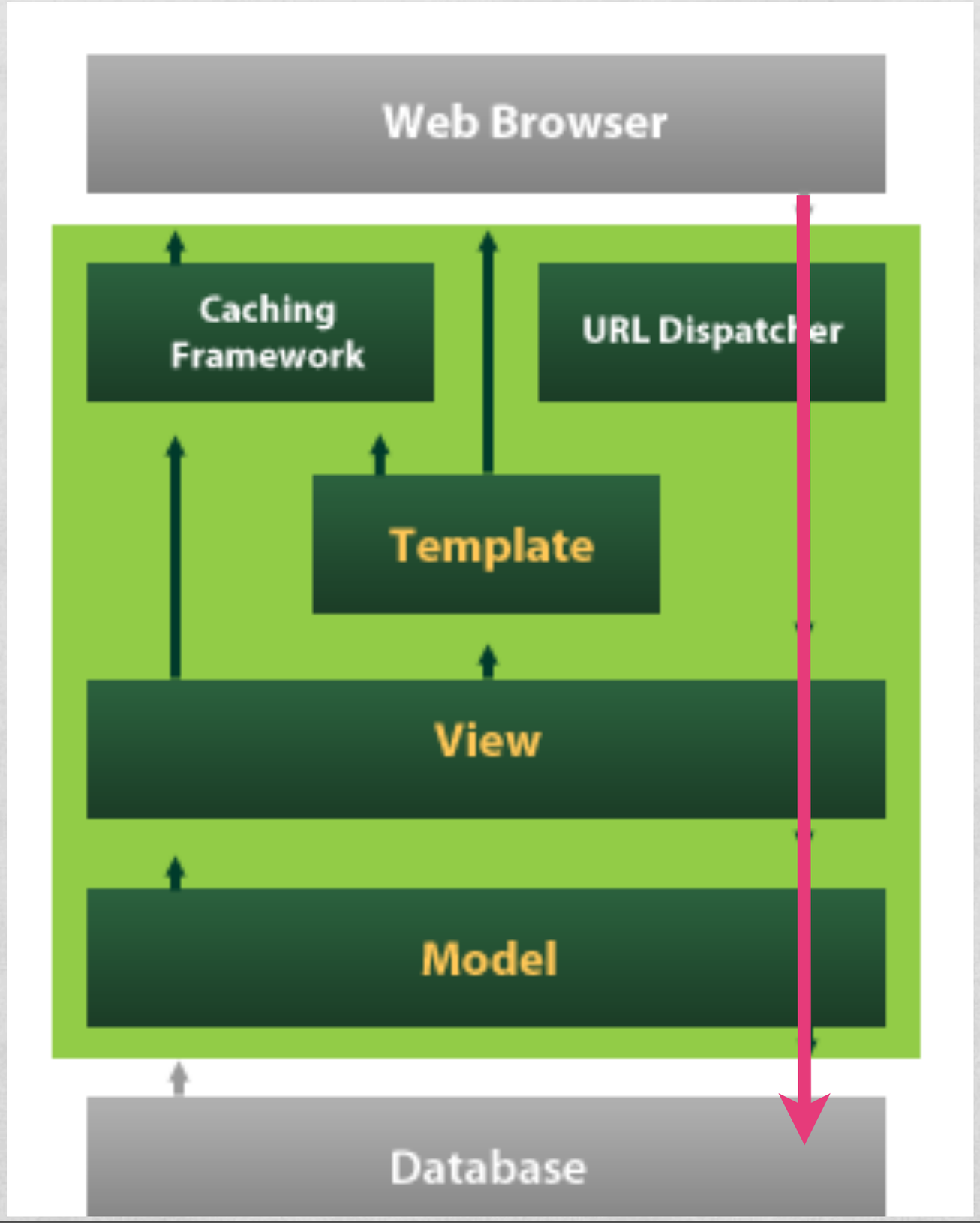
These are all database fields

DJANGO TEMPLATES

```
{% block content %}

    {{ category.name }}

    <h1>Articles</h1>

{% for article in articles %}
    <div class="article">
        <h2>{{ article.title }}</h2>

        <div class="content">
             {{ article.content|markdown }}
             —written by {{ author }}
        </div>
    </div>
{% endfor %}
{% endblock %}
```

TEMPLATES - HTML BUT WITH PYTHON CODE

THE BIG PICTURE



LETS

TRY IT