An Introduction to Django Web Framework

Agenda

Introduction

django

History

Created at Lawrence Journal-World newspaper in 2003 in Kansas, USA

Adrian Holovaty and Simon Willison initially wrote from scratch as nothing fitted

Historically excellent documentation

Named after French Guitarist Django Reinhardt

Open sourced in 2005

What is Django?

A high-level Python Web framework that encourages rapid development and clean, pragmatic design

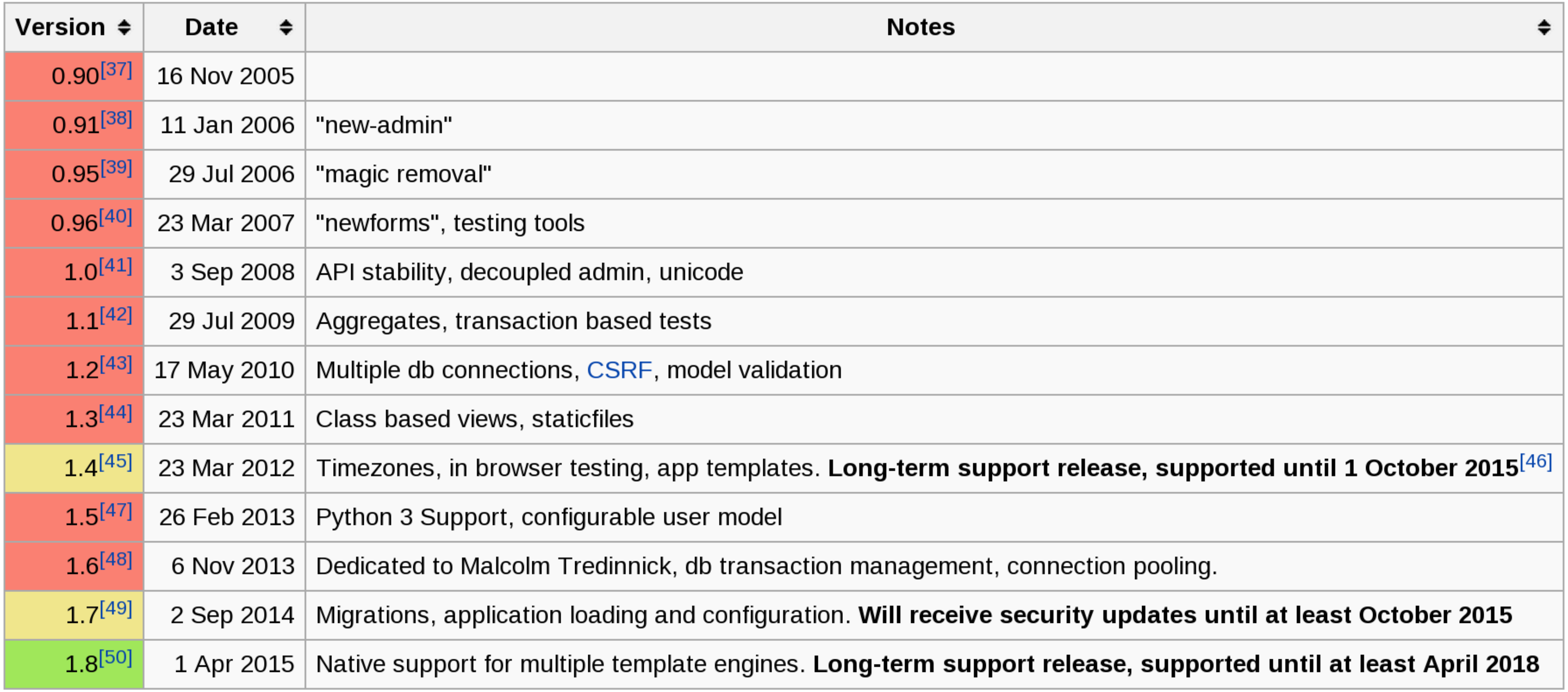
A free and open source web framework (public repository at https://github.com/django/django)

Model–View–Template (MVT) architectural pattern

Don’t Repeat Yourself (DRY) principle

Lots of out-of-the box features

Django Version History



Version	Date	Notes
0.90[37]	16 Nov 2005	
0.91[38]	11 Jan 2006	"new-admin"
0.95[39]	29 Jul 2006	"magic removal"
0.96[40]	23 Mar 2007	"newforms", testing tools
1.0[41]	3 Sep 2008	API stability, decoupled admin, unicode
1.1[42]	29 Jul 2009	Aggregates, transaction based tests
1.2[43]	17 May 2010	Multiple db connections, CSRF, model validation
1.3[44]	23 Mar 2011	Class based views, staticfiles
1.4[45]	23 Mar 2012	Timezones, in browser testing, app templates. **Long-term support release, supported until 1 October 2015**[46]
1.5[47]	26 Feb 2013	Python 3 Support, configurable user model
1.6[48]	6 Nov 2013	Dedicated to Malcolm Tredinnick, db transaction management, connection pooling.
1.7[49]	2 Sep 2014	Migrations, application loading and configuration. **Will receive security updates until at least October 2015**
1.8[50]	1 Apr 2015	Native support for multiple template engines. **Long-term support release, supported until at least April 2018**

What is a Web Framework?

Software framework designed to simplify your web development life

Handles many of the common operations associated with web development including:

- accessing the database
- creating HTML templates
- managing sessions

Promotes code reuse

Some other popular Python Web Frameworks

Flask: micro framework that provides a simple template for web development

Tornado: web framework and asynchronous networking library, noted for high performance

Bottle: fast, simple lightweight micro web-framework

Pyramid: MVT framework with flexible component choices

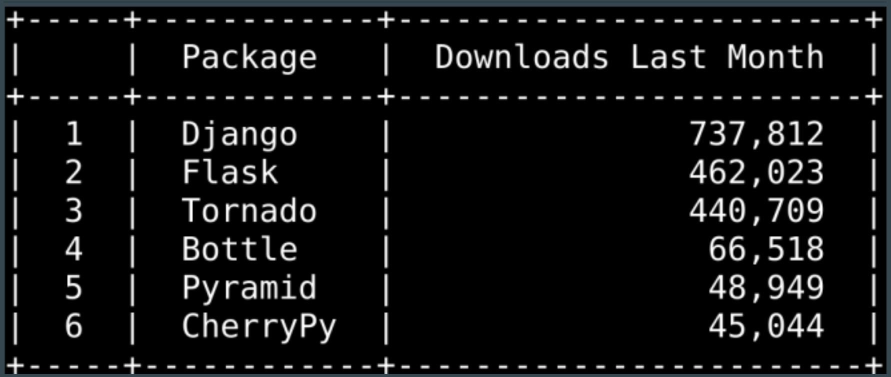
CherryPy: mature object-oriented web framework - compact and simple

How popular is Django?

PyPI is the official 3rd party package repository for Python

Stats available on number of downloads in last week, month, year.

Django is the most downloaded web framework, is regularly updated by community



```
+-----+------------+-----------------------+
|     |  Package   |  Downloads Last Month  |
+-----+------------+-----------------------+
|  1  |  Django    |               737,812  |
|  2  |  Flask     |               462,023  |
|  3  |  Tornado   |               440,709  |
|  4  |  Bottle    |                66,518  |
|  5  |  Pyramid   |                48,949  |
|  6  |  CherryPy  |                45,044  |
+-----+------------+-----------------------+
```

Websites using Django

Pinterest

Instagram

DISQUS

The Washington Post

Atlassian Bitbucket

the ONION

mozilla FOUNDATION

NATIONAL GEOGRAPHIC CHANNEL

NASA

Eventbrite

openstack™

theguardian

Getting Started

Polls Web Application

Build a web application to allow the public to vote in polls, and view results

Typical requirement for a online news website (such a *Lawrence Journal-World*)

Three interested parties:

- A software developer creates the necessary models
- A site administrator creates/updates/deletes the polls
- A public website visitor votes in the polls and views results

Same example application as the official Django tutorial, for more detailed explanation see https://docs.djangoproject.com/en/1.8/intro/tutorial01/

Django Quick Start Guide

1. Django needs to be installed!
2. Create a new project directory, install Django (preferable with pip and virtualenv)
3. To create Django project “mysite” run:

```
$ django-admin.py startproject mysite
```

Creates a directory structure which is the base of project:

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

4. Django supports 4 databases:

PostgreSQL MySQL SQLite ORACLE

Update the settings.py file to use sqlite and ‘Europe/Dublin’.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

TIME_ZONE = 'Europe/Dublin'
```

5. To create the sqlite file and database tables initially needed by Django run:

```
$ python manage.py migrate
```

6. Run the built-in Django development server (default port 8000):

```
$ python manage.py runserver
```

7. Open 127.0.0.1:8000 in web browser, and you should see a page saying “It worked!”

Application Quick Start Guide

8. A Django “project” is a collection of Django “app”s.

   An app is an application that does something specific (blog or poll)

   To start our new polls app, cd into ‘mysite’ and run:

```
$ django-admin.py startapp polls
```

   This creates the directory structure for your ‘polls’ app:

```
polls/
    __init__.py
    admin.py
    migrations/
        __init__.py
    models.py
    tests.py
    views.py
```

9. Add polls application to INSTALLED_APPS in settings.py to activate ‘polls’ models

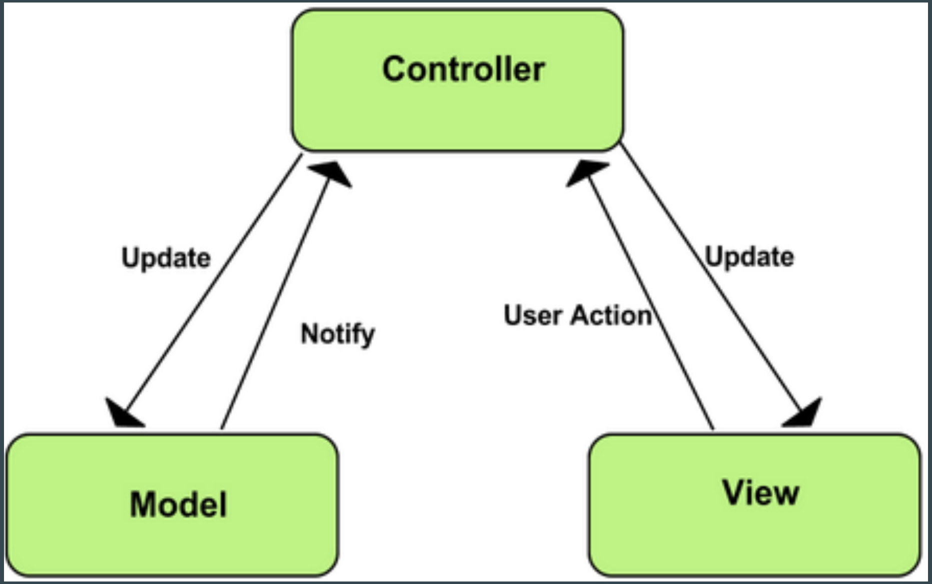
```
INSTALLED_APPS = (
    ...
    ‘polls’
)
```

10. Start working on the models, views, urls and templates for the polls app!

Django Architecture

What is MVC (Model-View-Controller)?

Software architectural pattern for user interfaces, popular in web applications.



Other MVC frameworks: Ruby-on-rails (Ruby), ASP.NET MVC (C# or Visual Basic)

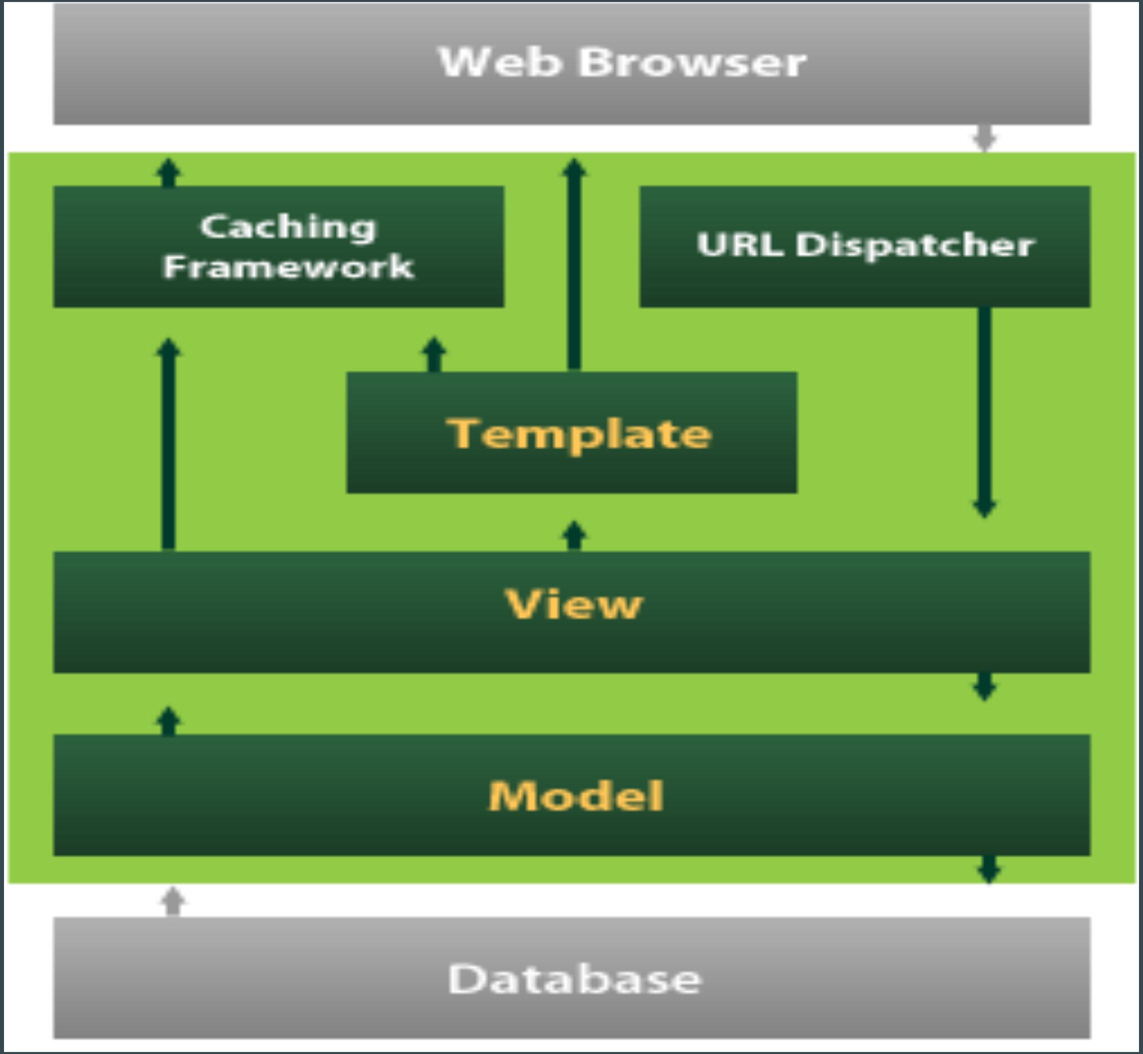
Alternative Django-specific acronym is MTV (Model, Template, View)

Template - User facing

URLs - URL Configuration

View - Business Logic

Model - Database Table



Models

Single definitive source of information about data

Define the interface to the database, an auto-generated database API

Generally maps to a single database table, and each attribute represents a database field

```
from django.db import models

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')
```

Models (continued)

Once the models are defined, two commands are run to create the database tables:

```
$ python manage.py makemigrations polls
```

```
$ python manage.py migrate polls
```

Views

Python function which takes a Web request, and returns a Web response

This response can be HTML for a web page, a redirect, a 404 Not Found error etc.

Contains the logic on what is returned from a particular request

Function-based or class-based

```
from django.http import HttpResponse
from .models import Question

def index(request):
    latest_question_list = Question.objects.order_by('-pub_date')[:5]
    output = ‘. ’.join([q.question_text for q in latest_question_list])
    return HttpResponse(output)
```

Views (continued)

Often required to return some variables from the model to a HTML template. The useful ‘render’ shortcut:

```
from django.shortcuts import render
from .models import Question

def index(request):
    latest_question_list = Question.objects.order_by('-pub_date')[:5]
    context = {'latest_question_list': latest_question_list}
    return render(request, ‘polls/index.html’, context)
```

Views are also used to check that a user is logged in, and to handle HTTP GET and POST requests (request.GET, request.POST)

URLs

Map a user request to the appropriate Django ‘*view*’

When accessed on the site, regex matching is performed on the path.

If no regex matched, then error handling is done.

Designed to make URLs elegant and understandable

A urls.py file for the base project and an urls.py file for each app.

URLs (continued)

*<domain_name>/**polls/** -* directs the request to the views.index function

*<domain_name>/**polls/4/** -* directs the request to the views.detail function with argument 4

```
from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^$', views.index, name='index'),
    url(r'^(?P<question_id>[0-9]+)/$', views.detail, name='detail')
]
```

Optional “name” argument to reference a URL

Templates

Simply a text file. It can generate any-text based format (HTML, XML, CSV, etc.).

Site-wide templates live in a “*templates*” directory on the project base

App-specific templates live within the app ‘polls/templates’

Templates encourage separation of application and presentation logic (no Python knowledge is required for templates)

Django has its own template engine and also supports Jinja2 engine

Templates (continued)

Contains variables “{{ }}”, which get replaced when the template is evaluated, and tags “{% %}” which control the logic of the template

```
<ul>
{% for question in latest_question_list %}
    <li><a href="{% url 'polls:detail' question.id %}">{{
question.question_text }}</a></li>
{% endfor %}
</ul>
```

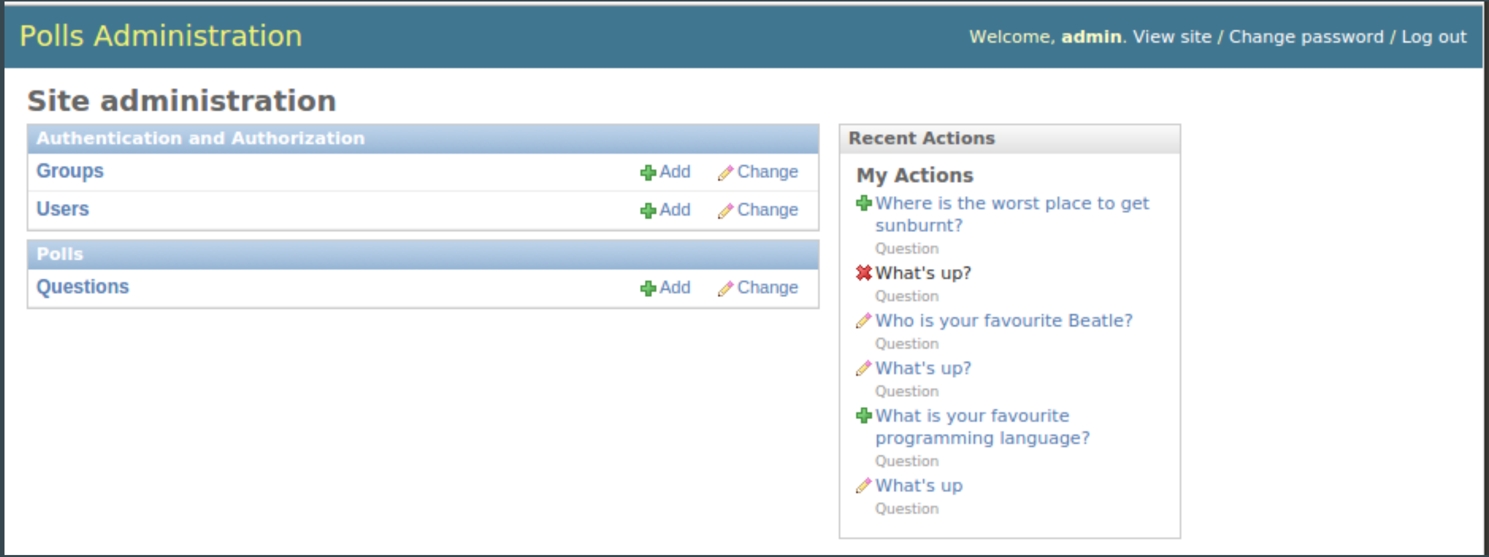
Template inheritance: A base HTML template can be used site-wide, and extended in each template

Other components

Admin

Useful auto-generated Django admin, which contains generated forms.

Should only be accessible by a trusted staff member to Create, Update, Delete site content.


Polls Administration

Welcome, **admin**. View site / Change password / Log out

Site administration

Authentication and Authorization		
Groups	Add	Change
Users	Add	Change

Polls		
Questions	Add	Change

Recent Actions

My Actions

- Where is the worst place to get sunburnt? — Question
- What's up? — Question
- Who is your favourite Beatle? — Question
- What's up? — Question
- What is your favourite programming language? — Question
- What's up — Question

Admin (continued)

To use the admin, first create a superuser (with username and password):

```
$ python manage.py createsuperuser
```

Once done, login to the admin site at: “<domain_name>/admin/”

Use admin.py files in each app to decide what and how to display model content to the staff user.

```
from django.contrib import admin
from .models import Question

class QuestionAdmin(admin.ModelAdmin):
    fields = [‘pub_date’, ‘question_text’]

admin.site.register(Question, QuestionAdmin)
```

Static Files

Includes images, JavaScript and CSS

Create a new directory ‘static’ within your app for these files ‘polls/static’

If using static files outside of an app, set STATICFILES_DIR in the settings file

Use the ‘static’ template tag in templates to avoid hardcoding paths to static files:

```
{% load staticfiles %}
<link rel="stylesheet" type="text/css" href="{% static 'polls/style.css' %}" />
```

Testing

“Code without tests is broken by design”

- Jacob Kaplan-Moss, one of original Django developers

To run the test cases for an app run:

```
$ python manage.py test polls
```

By default test runner will find any file named “*test*.py*” in current working directory

To run tests requiring database access subclass “*django.test.TestCase*”, a subclass of “unittest.TestCase” from the Python standard library

Testing (continued)

A very useful Test client which acts as a dummy Web browser can:

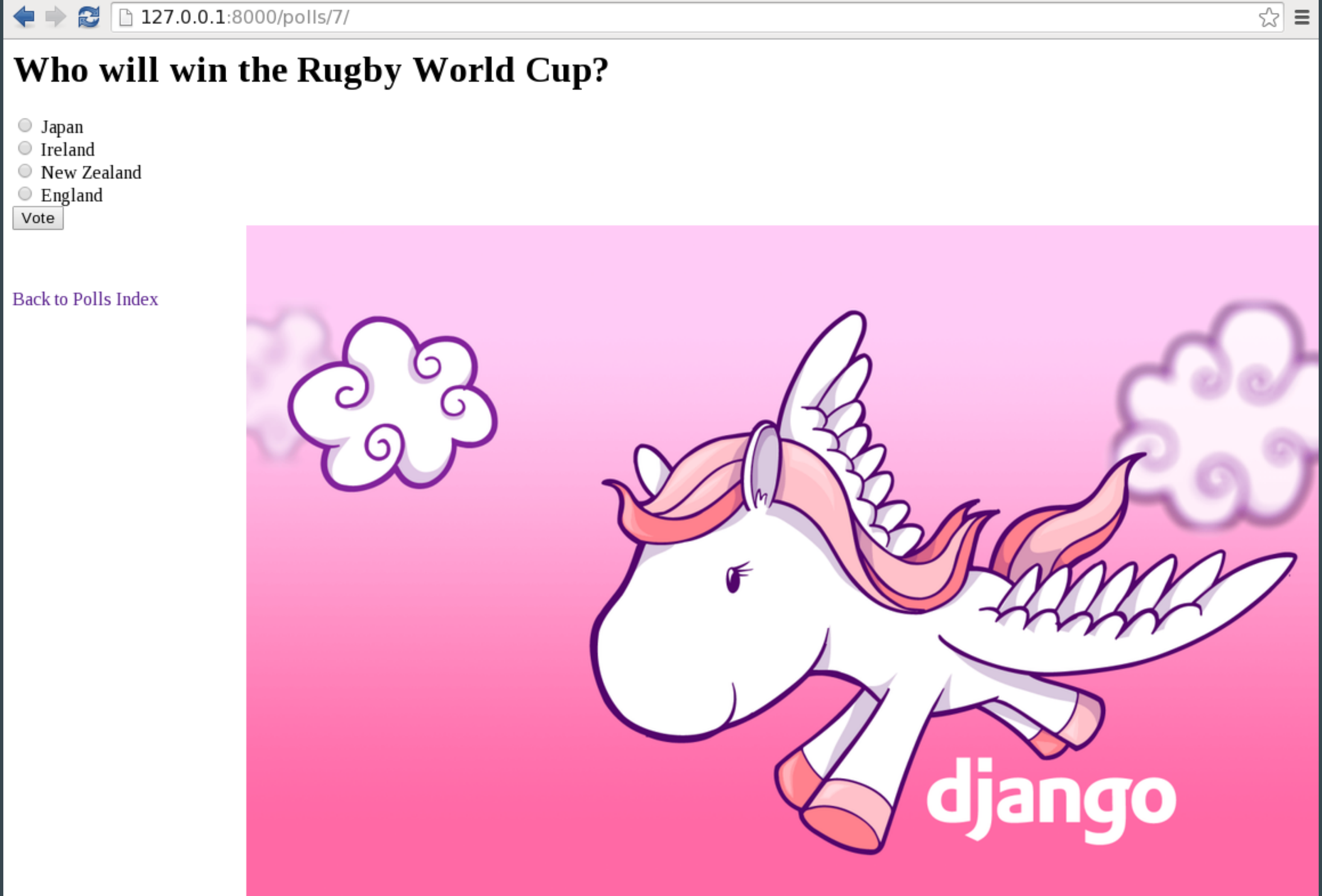
- Simulate GET and POST request on a URL
- Check redirects are working
- Test that a given request is rendered by a certain template

```
from django.test import TestCase

class QuestionViewTest(TestCase):
    def test_index_view_with_no_questions(self):
        """
        If no questions exist, an appropriate message should be displayed
        """
        response = self.client.get(reverse('polls:index'))
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, "No polls are available.")
```

Demo

```
[david.gibbons@7V0DZY1 tech_talk]$ tree -I '*.pyc|*~' mysite/
mysite/
|-- db.sqlite3
|-- manage.py
|-- mysite
|   |-- __init__.py
|   |-- settings.py
|   |-- urls.py
|   `-- wsgi.py
|-- polls
|   |-- admin.py
|   |-- __init__.py
|   |-- migrations
|   |   |-- 0001_initial.py
|   |   `-- __init__.py
|   |-- models.py
|   |-- static
|   |   `-- polls
|   |       |-- images
|   |       |   `-- magic-pony-django-wallpaper.png
|   |       `-- style.css
|   |-- templates
|   |   `-- polls
|   |       |-- detail.html
|   |       |-- index.html
|   |       `-- results.html
|   |-- tests.py
|   |-- urls.py
|   `-- views.py
`-- templates
    `-- admin
        `-- base_site.html
```



127.0.0.1:8000/polls/7/

Who will win the Rugby World Cup?

- ○ Japan
- ○ Ireland
- ○ New Zealand
- ○ England

Vote

Back to Polls Index

3rd Party Packages

Often prefixed - “django-”:

- djangorestframework
- django-registration-redux
- django-admin-honeypot
- django-debug-toolbar
- django-storages

Many more can be searched for at www.djangopackages.com

Summary

Started at *Lawrence Journal-World* newspaper in Kansas, USA

Most downloaded Python web framework, powering many popular websites in existence today

Architecture is based around models, views, URLs and templates

Other useful “out-of-the-box” components are the admin interface, static file handling, testing and user management

Excellent documentation - learn by example

References and Useful Links

- https://docs.djangoproject.com/en/1.8/
- https://en.wikipedia.org/wiki/Django_(web_framework)
- https://wiki.python.org/moin/WebFrameworks
- http://twoscoopspress.org/