Webdev: Building Django Apps

Ryan Fox
Andrew Glassman
MKE Python

What Django is

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

The Web framework for perfectionists with deadlines.

What Django isn't

Lightweight

Wordpress

React/Angular/Vue.js

Who uses it

https://disqus.com/

https://www.instagram.com/

https://www.pinterest.com/

Many others!

Getting Django

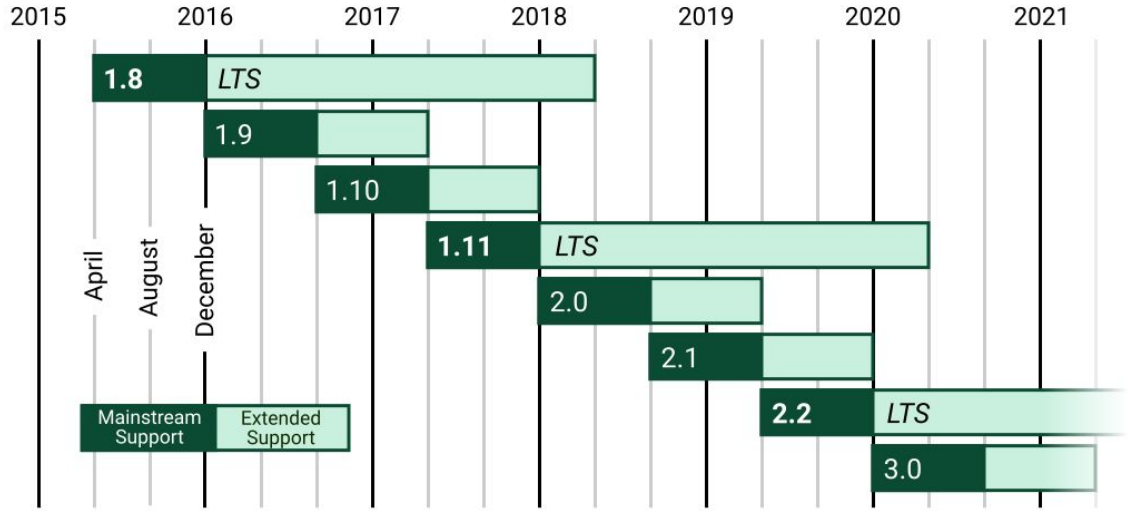
```
$ pip install django
```

https://github.com/django/django

https://www.djangoproject.com/

Getting Django

Use 1.11 now, 2.0 coming soon



Structure of Django

Models

Views

Templates

Structure of a Django project

```
$ django-admin startproject my_project

my_project/
├── manage.py
└── my_project
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

Structure of a Django project

```
$ python manage.py startapp my_app

my_project/my_app/
├── admin.py
├── apps.py
├── __init__.py
├── migrations
│   └── __init__.py
├── models.py
├── tests.py
└── views.py
```

URLs in Django

```
# urls.py
from django.conf.urls import url

from . import views

urlpatterns = [
    # ex: /polls/
    url(r'^$', views.index, name='index'),
    # ex: /polls/5/
    url(r'^(?P<question_id>[0-9]+)/$', views.detail, name='detail'),
    # ex: /polls/5/results/
    url(r'^(?P<question_id>[0-9]+)/results/$', views.results, name='results'),
    # ex: /polls/5/vote/
    url(r'^(?P<question_id>[0-9]+)/vote/$', views.vote, name='vote'),
]
```

URLs in Django

```
urlpatterns = [
    url(r'^$', IndexView.as_view(), name='index'),
    url(r'^about/$', AboutView.as_view(), name='about'),
    url(r'^faq/$', FaqView.as_view(), name='faq'),
    url(r'^getting-started/$', GettingStartedView.as_view(), name='getting_started'),
    url(r'^account/$', AccountView.as_view(), name='account'),
    url(r'^account/cancel/$', AccountCancelView.as_view(), name='account_cancel'),
    url(r'^dashboard/$', DashboardView.as_view(), name='dashboard'),
    url(r'^team/$', TeamView.as_view(), name='team'),
    url(r'^team/records/$', TeamRecordsView.as_view(), name='team_records'),
...
```

Structure of Django

Models

Views

Templates

Models

Corresponds to a “thing” in your application

Each model has a DB table

Each instance is one row in the DB

Models

```
# models.py
from django.db import models

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

class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

Model fields

Determine data type of each attribute on a model: String, int, datetime, etc...

Can specify constraints on each field: unique, not null, max length, ...

Model field types

BooleanField
CharField
DateField
DateTimeField
DurationField
EmailField
FileField
FloatField
ImageField

IntegerField
PositiveIntegerField
TextField
TimeField
URLField
UUIDField
ForeignKey*
ManyToManyField*
OneToOneField*

Models

```
# models.py
from django.db import models

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

class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    choice_text = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

Querying models

```
>>> from blog.models import Blog, Entry

>>> all_blogs = Blog.objects.all()

>>> all_entries = Entry.objects.all()
```

Querying models

```
>>> old_entries = Entry.objects.filter(pub_date__year=2006)

>>> january_entries = old_entries.filter(pub_date__month=1)

>>> new_entries = Entry.objects.filter(pub_date__gte=datetime.date(2017, 1, 1))
```

Querying models

```
>>> first_entry = Entry.objects.get(pk=1)
```

Structure of Django

Models

Views

Templates

Views

```
# views.py

from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello, world. You're at the polls index.")
```

Views

```
# views.py
from django.http import HttpResponse
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/polls.html', context)
```

Structure of Django

Models

Views

Templates

Templates

Skeletons that gets transformed to HTML

Composable - can include or be included in sub-templates

HTML + Django Template Language

Templates

```
<!DOCTYPE html>
<html lang="en">
<head>
     <title>{% block title %}Default title - can be overridden{% endblock %}</title>
     <link href="{% static 'css/main.css' %}" rel="stylesheet">
</head>

<body>
{% block content %}{% endblock %}
</body>
</html>
```

Templates

```
<!- polls.html ->
{% extends 'index.html' %}

{% block content %}
     {% if latest_question_list %}
         <ul>
         {% for question in latest_question_list %}
             <li><a href="/polls/{{ question.id }}/">{{ question.question_text }}</a></li>
         {% endfor %}
         </ul>
     {% else %}
         <p>No polls are available.</p>
     {% endif %}
{% endblock %}
```

Forms

Allow users to submit information

Many form fields correspond to model fields

Forms

```
<form action="/your-name/" method="post">
    <label for="your_name">Your name: </label>
    <input id="your_name" type="text" name="your_name" value="{{ current_name }}">
    <input type="submit" value="OK">
</form>
```

Creating forms

```
# forms.py

from django import forms

class NameForm(forms.Form):
    first_name = forms.CharField(label='Your first name', max_length=100)
    last_name = forms.CharField(label='Your last name', max_length=100)
```

Rendering forms

```
<!- submit.html ->

<form action="/your-name/" method="post">
    {% csrf_token %}
    {{ form }}
    <input type="submit" value="Submit" />
</form>
```

Using forms

```
# views.py
from django.shortcuts import render
from django.http import HttpResponseRedirect
from .forms import NameForm

def get_name(request):
    if request.method == 'POST':
        form = NameForm(request.POST)
        if form.is_valid():
            new_person = Person(form.cleaned_data[‘first_name’], form.cleaned_data[‘last_name’])
            new_person.save()
            return HttpResponseRedirect('/thanks/')
    else:
        form = NameForm()
    return render(request, 'name.html', {'form': form})
```

So much more

Migrations

Channels (websockets)

GIS

Caching

Deployment strategies

Tons more...

Learning more

https://docs.djangoproject.com/en/dev/intro/

https://djangocon.us/

https://www.twoscoopspress.com/products/two-scoops-of-django-1-11

http://www.django-rest-framework.org/

6000+ packages on PyPI

Thanks!

ryan@foxrow.com

https://www.meetup.com/MKE-Python-Meetup/