Django

Django is a widely-used Python web application framework with a "batteries-included" philosophy. The principle behind batteries-included is that the common functionality for building web applications should come with the framework instead of as separate libraries.

For example, authentication, URL routing, a template engine, an object-relational mapper (ORM), and database schema migrations are all included with the Django framework. Compare that included functionality to the Flask framework which requires a separate library such as Flask-Login to perform user authentication.

The batteries-included and extensibility philosophies are simply two different ways to tackle framework building. Neither philosophy is inherently better than the other one.

Django is an implementation of the web frameworks concept. Learn how these pieces fit together in the web development chapter or view the table of contents for all topics.

Why is Django a good web framework choice?

The Django project's stability, performance and community have grown tremendously over the past decade since the framework's creation. Detailed tutorials and good practices are readily available on the web and in books. The framework continues to add significant new functionality such as database migrations with each release.

I highly recommend the Django framework as a starting place for new Python web developers because the official documentation and tutorials are some of the best anywhere in software development. Many cities also have Django-specific groups such as Django District, Django Boston and San Francisco Django so new developers can get help when they are stuck.

Django books and tutorials

There are a slew of free or low cost resources out there for Django. Make sure to check the version numbers used in each post you read because Django was released over 10 years ago and has had a huge number of updates since then. These resources are geared towards beginners. If you are already experienced with Django you should take a look at the next section of resources for more advanced tutorials.

- Tango with Django is an extensive set of free introductions to using the most popular Python web framework. Several current developers said this book really helped them get over the initial framework learning curve.
- The Django Girls Tutorial is a great tutorial that doesn't assume any prior knowledge of Python or Django while helping you build your first web application.
- A Complete Beginner's Guide to Django is a wonderful seven-part series that incrementally builds out a Django project and handles deploying the app in the final post. The seven parts are:
  - Getting Started
  - Fundamentals
  - Advanced Concepts
  - Authentication
  - Django ORM
  - Class-Based Views
  - Deployment
- Test-Driven Development with Python focuses on web development using Django and JavaScript. This book uses the development of a website using the Django web framework as a real world example of how to perform test-driven development (TDD). There is also coverage of NoSQL, WebSockets and asynchronous responses. The book can be read online for free or purchased in hard copy via O'Reilly.
- Django OverIQ is a project-based tutorial for beginners that covers the required features such as the Django ORM and Django Templates.
- The Django subreddit often has links to the latest resources for learning Django and is also a good spot to ask questions about it.
- This Django tutorial shows how to build a project from scratch using Twitter Bootstrap, Bower, Requests and the Github API.
- The recommended Django project layout is helpful for developers new to Django to understand how to structure the directories and files within apps for projects.
- Django for Beginners: Build websites with Python and Django by William S. Vincent is perfect if you are just getting started with Django and web development, taking you from total beginner to confident web developer with Django and Python.

Django videos

Are you looking for Django videos in addition to articles? There is a special section for Django and web development on the best Python videos page.

Intermediate and advanced Django topics

These books and tutorials assume that you know the basics of building Django and want to go further to become much more knowledgeable about the framework.

- 2 Scoops of Django by Daniel Greenfeld and Audrey Roy is well worth the price of admission if you're serious about learning how to correctly develop Django websites.
- The Test-Driven Development with Django, Django REST Framework, and Docker course details how to set up a development environment with Docker in order to build and deploy a RESTful API powered by Python, Django, and Django REST Framework.
- User Interaction With Forms explains general web form input, how Django handles forms via POST requests, different types of input such as CharFields, DateFields and EmailFields, and validating that input.
- This 3-part Django project optimization guide covers a wide range of advanced topics such as Profiling and Django settings, working with databases and caching.
- Caching in Django is a detailed look at the configuration required for caching and how to measure the performance improvements once you have it in place.
- Mental Models for Class Based Views provides some comparison points between class based views (CBVs) and function based views and the author's opinions for how you can better understand CBVs.
- Working with time zones is necessary for every web application. This blog post on pytz and Django is a great start for figuring out what you need to know.
- A Guide to ASGI in Django 3.0 and its Performance covers the new Asynchronous Server Gateway Interface (ASGI) that was introduced in Django 3.0 and explains some of the nuances and gotchas that you should consider if you decide to use it for your web apps.
- REST APIs with Django: Build powerful web APIs with Python and Django by William S. Vincent is the book for you if you are just moving beyond the basics of Django and looking to get up speed with Django REST Framework (DRF) and service-oriented architecture (SOA). It also dives into more advanced topics like token-based authentication and permissions.
- Django Stripe Tutorial details how to quickly add Stripe to accept payments in a Django web app.
- This Python Social Auth for Django tutorial will show you how to integrate social media sign in buttons into your Django application.
- Upgrading Django provides a version-by-version guide for updating your Django projects' code.
- The Django Admin Cookbook and Building Multi Tenant Applications with Django are two solid "code recipe-style" free open source books that will teach you more about the admin interface as well as building projects that will be used by more than a single customer so their data needs to be properly separated.
- How to Create Custom Django Management Commands explains how to expand the default `manage.py` commands list with your own custom commands in your projects. The tutorial has a bunch of great examples with expected output to make it easy to follow along and learn while you work through the post.
- Luke Plant writes about his approach to class based views (CBVs), which often provoke heated debate in the Django community for whether they are a time saver or "too much magic" for the framework.
- Django Apps Checklist gives some good practices rules for building reusable Django apps.

Django migrations

- Paul Hallett wrote a detailed Django 1.7 app upgrade guide on the Twilio blog from his experience working with the django-twilio package.
- Real Python's migrations primer explores the difference between South's migrations and the built-in Django 1.7 migrations as well as how you use them.
- Andrew Pinkham's "Upgrading to Django 1.7" series is great learning material for understanding what's changed in this major release and how to adapt your Django project. Part 1, part 2 and part 3 and part 4 are now all available to read.
- Django migrations without downtimes shows one potential way of performing on-line schema migrations with Django.
- How to Extend Django User Model presents four main ways to expand upon the built-in `User` model that is packaged with Django. This scenario is very common for all but the simplest Django projects.
- Creating a Custom User Model in Django looks at how to create a custom User model in Django so that an email address can be used as the primary user identifier instead of a username for authentication.

Django Channels

Channels are a new mechanism in Django 1.9 provided as a standalone app. They may be incorporated into the core framework in 2.0+. Channels provide "real-time" full-duplex communication between the browser and the server based on WebSockets.

- This tutorial shows how to get started with Django Channels in your project.
- The channels examples repository contains a couple of good starter projects such as a live blog and a chat application to use as base code.
- The Developing a Real-Time Taxi App with Django Channels and Angular course details how to create a ride-sharing app with Django Channels, Angular, and Docker. Along the way, you'll learn how to manage client/server communication with Django Channels, control flow and routing with Angular, and build a RESTful API with Django REST Framework.

Django testing

- Integrating Front End Tools with Django is a good post to read for figuring out how to use Gulp for handling front end tools in development and production Django sites.
- Django Testing Cheat Sheet covers many common scenarios for Django applications such as testing POST requests, request headers, authentication, and large numbers of model fields in the Django ORM.
- Getting Started with Django Testing will help you stop procrastinating on testing your Django projects if you're uncertain where to begin.
- Testing in Django provides numerous examples and explanations for how to test your Django project's code.
- Django views automated testing with Selenium gives some example code to get up and running with Selenium browser-based tests.

Django with JavaScript MVC frameworks

There are resources for JavaScript MVC frameworks such as Angular, React and Vue.js on their respective pages.

Django ORM tutorials

Django comes with its own custom object-relational mapper (ORM) typically referred to as "the Django ORM". Learn more about the Django ORM on the its own page and more broadly about ORMs on the Python object-relational mappers page.

Static and media files

Deploying and handling static and media files can be confusing for new Django developers. These resources along with the static content page are useful for figuring out how to handle these files properly.

- Using Amazon S3 to Store your Django Site's Static and Media Files is a well written guide to a question commonly asked about static and media file serving.
- Loading Django FileField and ImageFields from the file system shows how to load a model field with a file from the file system.
- Storing Django Static and Media Files on Amazon S3 shows how to configure Django to load and serve up static and media files, public and private, via an Amazon S3 bucket.

Django project templates

Project templates, not to be confused with a template engine, generate boilerplate code for a base Django project plus optional libraries that are often used when developing web applications.

- Caktus Group's Django project template is Django 2.2+ ready.
- Cookiecutter Django is a project template from Daniel Greenfeld, for use with Audrey Roy's Cookiecutter. The template results are Heroku deployment-ready.
- Two Scoops Django project template is also from the PyDanny and Audrey Roy. This one provides a quick scaffold described in the Two Scoops of Django book.
- Sugardough is a Django project template from Mozilla that is compatible with cookiecutter.

Open source Django example projects

Reading open source code can be useful when you are trying to figure out how to build your own projects. This is a short list of some real-world example applications, and many more can be found on the Django example projects and code page.

- Openduty is a website status checking and alert system similar to PagerDuty.
- Courtside is a pick up sports web application written and maintained by the author of PyCoder's Weekly.
- These two Django Interactive Voice Response (IVR) system web application repositories part 1 and part 2 show you how to build a really cool Django application. There's also an accompanying blog post with detailed explanations of each step.
- Taiga is a project management tool built with Django as the backend and AngularJS as the front end.
- Chowist is a web application that replicates core features of Yelp and adds a couple more bells and whistles.

Open source code to learn Django

There are many open source projects that rely on Django. One of the best ways to learn how to use this framework is to read how other projects use it in real-world code. This section lists these code examples by class and method in Django's code base.

Django: Extensions, Plug-ins and Related Libraries & Example Projects and Code

django.apps.config AppConfig

django.conf settings, urls.url

django.contrib.admin filters.SimpleListFilter,

django.contrib.admin.helpers ActionForm, AdminForm

django.contrib.admin.options IS_POPUP_VAR, IncorrectLookupParameters, ModelAdmin, csrf_protect_m

django.contrib.admin.sites NotRegistered, register, site

django.contrib.auth get_user_model, decorators login_required, hashers make_password

django.contrib.staticfiles finders, .finders BaseFinder, .finders BaseStorageFinder, .finders find, .finders get_finders .handlers StaticFilesHandler storage .storage CachedStaticFilesStorage, .storage HashedFilesMixin, .storage ManifestStaticFilesStorage, .storage StaticFilesStorage, .storage staticfiles_storage .utils matches_patterns

django.core cache, checks, exceptions, mail, mail.send_mail, mail.messages EmailMessage, management, management.base BaseCommand, serializers, signals, signing, validators

django.core.exceptions DisallowedRedirect, FieldDoesNotExist, FieldError, ImproperlyConfigured, MiddlewareNotUsed, NON_FIELD_ERRORS, ObjectDoesNotExist, PermissionDenied, SuspiciousFileOperation, SuspiciousMultipartForm, ValidationError

django.db DEFAULT_DB_ALIAS, DataError, DatabaseError, IntegrityError, OperationalError, ProgrammingError, connection, connections, migrations, router, transaction .backends utils

django.db.migrations RunPython, .autodetector MigrationAutodetector, .exceptions IrreversibleError, .executor MigrationExecutor, .loader MIGRATIONS_MODULE_NAME, .loader MigrationLoader, .operations.base Operation, .state ProjectState

django.db.models AutoField, BooleanField, CharField, DateField, DateTimeField, FileField, ForeignKey, GenericIPAddressField, ImageField, IntegerField, Model, PositiveIntegerField, PositiveSmallIntegerField, signal, SlugField, SmallIntegerField, TextField

django.db.models.query BaseIterable, EmptyQuerySet, ModelIterable, Prefetch, Q, QuerySet, prefetch_related_objects

django.db.models.query_utils DeferredAttribute, PathInfo, Q

django.db.models.signals post_delete, post_save, pre_delete, pre_save

django.dispatch.dispatcher Signal

django.forms BaseForm, BooleanField, CharField, CheckboxInput, CheckboxSelectMultiple, ChoiceField, DateField, DateInput, DateTimeField, EmailField, Field, FileInput, FilePathField, Form, HiddenInput, ImageField, IntegerField, Media, MediaDefiningClass, ModelChoiceField, ModelForm, ModelMultipleChoiceField, MultipleChoiceField, Select, SelectMultiple, TypedChoiceField, ValidationError

django.http HttpResponse, HttpResponseBadRequest, HttpResponseForbidden, HttpResponseNotModified, Http404, HttpResponsePermanentRedirect, HttpResponseRedirect

django.shortcuts get_list_or_404, get_object_or_404, redirect, render, resolve_url

django.template.base Context, FilterExpression, Node, NodeList, Parser, Template, TemplateSyntaxError, TextNode, Token, TokenType, VariableDoesNotExist, VariableNode, token_kwargs

django.template.context Context

django.template.defaultfilters escape, filesizeformat, safe, slugify, striptags, title, truncatechars

django.template.loader get_template, render_to_string, select_template

django.template.loader_tags BlockNode, ExtendsNode, IncludeNode

django.template.loaders.filesystem Loader

django.template.response SimpleTemplateResponse, TemplateResponse

django.urls URLPattern, URLResolver, clear_url_caches, get_callable, get_resolver, get_script_prefix, include, path, re_path, register_converter, resolve, reverse, reverse_lazy

django.urls.exceptions NoReverseMatch, Resolver404

django.utils dateformat, dateparse, datetime_safe, formats, module_loading, termcolors, timezone, translation, tree

django.utils.cache add_never_cache_headers, cc_delim_re, patch_cache_control, patch_response_headers, patch_vary_headers

django.utils.crypto constant_time_compare, get_random_string

django.utils.datastructures MultiValueDict

django.utils.dateparse parse_datetime, parse_duration

django.utils.dates MONTHS

django.utils.datetime_safe datetime

django.utils.decorators method_decorator

django.utils.deprecation MiddlewareMixin, RenameMethodsBase

django.utils.duration duration_string

django.utils.encoding DjangoUnicodeDecodeError, filepath_to_uri, force_bytes, force_str, force_text, iri_to_uri, is_protected_type, smart_bytes, smart_str, smart_text, uri_to_iri

django.utils.formats get_format, localize_input, sanitize_separators

django.utils.functional LazyObject, Promise, SimpleLazyObject, keep_lazy, lazy, total_ordering, wraps

django.utils.html conditional_escape, escape, escapejs, format_html, format_html_join, mark_safe, smart_urlquote, strip_tags

django.utils.http base36_to_int, http_date, int_to_base36, is_safe_url, unquote, url_has_allowed_host_and_scheme, urlencode, urlquote, urlunquote

django.utils.ipv6 clean_ipv6_address

django.utils.itercompat is_iterable

django.utils.module_loading autodiscover_modules, import_string, module_has_submodule

django.utils.numberformat format

django.utils.safestring SafeData, SafeText, mark_safe

django.utils.termcolors colorize

django.utils.text Truncator, capfirst, format_lazy, get_text_list, get_valid_filename, slugify

django.utils.timezone get_current_timezone, make_aware, now, timedelta

django.utils.translation LANGUAGE_SESSION_KEY, activate, deactivate_all, get_language, get_language_from_request, gettext, gettext_lazy, ngettext, override, pgettext, pgettext_lazy, ugettext, ugettext_lazy, ungettext, ungettext_lazy

django.utils.version get_complete_version

django.views csrf .debug get_default_exception_reporter_filter .decorators.csrf csrf_exempt .decorators.debug sensitive_post_parameters .decorators.http require_GET, .decorators.http require_POST

django.views.generic CreateView, DeleteView, DetailView, FormView, ListView, RedirectView, TemplateView, UpdateView, View

django.views.generic.base RedirectView, TemplateResponseMixin, TemplateView, View

django.views.generic.detail SingleObjectMixin

django.views.generic.edit CreateView, DeleteView, DeletionMixin, FormMixin, FormView

django.views.generic.list ListView, MultipleObjectMixin

django.views.i18n JavaScriptCatalog

django.views.static serve, was_modified_since

What do you need to learn next for your Django app?

My app runs but looks awful. How do I style the user interface?

How do I integrate existing web APIs into my application?

I've built a Python web app, now how do I deploy it?

Table of Contents

Full Stack Python

Full Stack Python is an open book that explains concepts in plain language and provides helpful resources for those topics.

Updates via Twitter & Facebook.