Django

Documentation

Django documentation

Everything you need to know about Django.

First steps

Are you new to Django or to programming? This is the place to start!

- **From scratch:** Overview | Installation
- **Tutorial:** Part 1: Requests and responses | Part 2: Models and the admin site | Part 3: Views and templates | Part 4: Forms and generic views | Part 5: Testing | Part 6: Static files | Part 7: Customizing the admin site
- **Advanced Tutorials:** How to write reusable apps | Writing your first patch for Django

Getting help

Having trouble? We'd like to help!

- Try the FAQ – it's got answers to many common questions.
- Looking for specific information? Try the Index, Module Index or the detailed table of contents.
- Not found anything? See FAQ: Getting Help for information on getting support and asking questions to the community.
- Report bugs with Django in our ticket tracker.

How the documentation is organized

Django has a lot of documentation. A high-level overview of how it's organized will help you know where to look for certain things:

- Tutorials take you by the hand through a series of steps to create a Web application. Start here if you're new to Django or Web application development. Also look at the "First steps".
- Topic guides discuss key topics and concepts at a fairly high level and provide useful background information and explanation.
- Reference guides contain technical reference for APIs and other aspects of Django's machinery. They describe how it works and how to use it but assume that you have a basic understanding of key concepts.
- How-to guides are recipes. They guide you through the steps involved in addressing key problems and use-cases. They are more advanced than tutorials and assume some knowledge of how Django works.

The model layer

Django provides an abstraction layer (the "models") for structuring and manipulating the data of your Web application. Learn more about it below:

- **Models:** Introduction to models | Field types | Indexes | Meta options | Model class
- **QuerySets:** Making queries | QuerySet method reference | Lookup expressions
- **Model instances:** Instance methods | Accessing related objects
- **Migrations:** Introduction to Migrations | Operations reference | SchemaEditor | Writing migrations
- **Advanced:** Managers | Raw SQL | Transactions | Aggregation | Search | Custom fields | Multiple databases | Custom lookups | Query Expressions | Conditional Expressions | Database Functions
- **Other:** Supported databases | Legacy databases | Providing initial data | Optimize database access | PostgreSQL specific features

Getting Help

Language: **en**

Documentation version: **3.2**

The view layer

Django has the concept of "views" to encapsulate the logic responsible for processing a user's request and for returning the response. Find all you need to know about views via the links below:

- **The basics:** URLconfs | View functions | Shortcuts | Decorators | Asynchronous Support

- **Reference:** Built-in Views | Request/response objects | TemplateResponse objects

- **File uploads:** Overview | File objects | Storage API | Managing files | Custom storage

- **Class-based views:** Overview | Built-in display views | Built-in editing views | Using mixins | API reference | Flattened index

- **Advanced:** Generating CSV | Generating PDF

- **Middleware:** Overview | Built-in middleware classes

The template layer

The template layer provides a designer-friendly syntax for rendering the information to be presented to the user. Learn how this syntax can be used by designers and how it can be extended by programmers:

- **The basics:** Overview

- **For designers:** Language overview | Built-in tags and filters | Humanization

- **For programmers:** Template API | Custom tags and filters | Custom template backend

Forms

Django provides a rich framework to facilitate the creation of forms and the manipulation of form data.

- **The basics:** Overview | Form API | Built-in fields | Built-in widgets

- **Advanced:** Forms for models | Integrating media | Formsets | Customizing validation

The development process

Learn about the various components and tools to help you in the development and testing of Django applications:

- **Settings:** Overview | Full list of settings

- **Applications:** Overview

- **Exceptions:** Overview

- **django-admin and manage.py:** Overview | Adding custom commands

- **Testing:** Introduction | Writing and running tests | Included testing tools | Advanced topics

- **Deployment:** Overview | WSGI servers | ASGI servers | Deploying static files | Tracking code errors by email | Deployment checklist

The admin

Find all you need to know about the automated admin interface, one of Django's most popular features:

- Admin site

- Admin actions

- Admin documentation generator

Security

Security is a topic of paramount importance in the development of Web applications and Django provides multiple protection tools and mechanisms:

- Security overview

- Disclosed security issues in Django

- Clickjacking protection

- Cross Site Request Forgery protection

- Cryptographic signing

- Security Middleware

Getting Help

Language: **en**

Documentation version: **3.2**

Internationalization and localization

Django offers a robust internationalization and localization framework to assist you in the development of applications for multiple languages and world regions:

- Overview | Internationalization | Localization | Localized Web UI formatting and form input

- Time zones

Performance and optimization

There are a variety of techniques and tools that can help get your code running more efficiently - faster, and using fewer system resources.

- Performance and optimization overview

Geographic framework

GeoDjango intends to be a world-class geographic Web framework. Its goal is to make it as easy as possible to build GIS Web applications and harness the power of spatially enabled data.

Common Web application tools

Django offers multiple tools commonly needed in the development of Web applications:

- **Authentication:** Overview | Using the authentication system | Password management | Customizing authentication | API Reference

- Caching

- Logging

- Sending emails

- Syndication feeds (RSS/Atom)

- Pagination

- Messages framework

- Serialization

- Sessions

- Sitemaps

- Static files management

- Data validation

Other core functionalities

Learn about some other core functionalities of the Django framework:

- Conditional content processing

- Content types and generic relations

- Flatpages

- Redirects

- Signals

- System check framework

- The sites framework

- Unicode in Django

Getting Help

Language: **en**

The Django open-source project

Documentation version: **3.2**

Learn about the development process for the Django project itself and about how you can contribute:

- **Community:** How to get involved | The release process | Team organization | The Django source code repository | Security policies | Mailing lists
- **Design philosophies:** Overview
- **Documentation:** About this documentation
- **Third-party distributions:** Overview
- **Django over time:** API stability | Release notes and upgrading instructions | Deprecation Timeline

Learn More

About Django

Getting Started with Django

Team Organization

Django Software Foundation

Code of Conduct

Diversity Statement

Get Involved

Join a Group

Contribute to Django

Submit a Bug

Report a Security Issue

Follow Us

GitHub

Twitter

News RSS

Django Users Mailing List

Getting Help

Support Us

Language: **en**

Sponsor Django

Documentation version: **3.2**

Official merchandise store

Amazon Smile

Benevity Workplace Giving Program

Getting Help

Language: **en**

Documentation version: **3.2**