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 | Part 8: Adding third-party packages
- **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

---

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