

Website Content Analysis Report

URL: <https://flask.palletsprojects.com>

Summary

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. There is also a more detailed Tutorial that shows how to create a small but complete application with Flask. Users Guide Flask provides configuration and conventions, with sensible defaults, to get started. This section of the documentation explains the different parts of the Flask framework and how they can be used, customized, and extended. Beyond Flask itself, look for community-maintained extensions to add even more functionality.

Installation Python Version Dependencies Virtual environments Install Flask Quickstart A Minimal Application Debug Mode HTML Escaping Routing Static Files Rendering Templates Accessing Request Data Redirects and Errors About Responses Sessions Message Flashing Logging Hooking in WSGI Middleware Using Flask Extensions Deploying to a Web Server Tutorial Project Layout Application Setup Define and Access the Database Blueprints and Views Templates Static Files Blog Blueprint Make the Project Installable Test Coverage Deploy to Production Keep Developing Templates Jinja Setup Standard Context Controlling Autoescaping Registering Filters Context Processors Streaming Testing Flask Applications Identifying Tests Fixtures Sending Requests with the Test Client Following Redirects Accessing and Modifying the Session Running Commands with the CLI Runner Tests that depend on an Active Context Handling Application Errors Error Logging Tools Error Handlers Custom Error Pages Blueprint Error Handlers Returning API Errors as JSON Logging Debugging Debugging Application Errors In Production The Builtin Debugger External Debuggers Logging Basic Configuration Email Errors to Admins Injecting Request Information Other Libraries Configuration Handling Configuration Basics Debug Mode Builtin Configuration Values Configuring from Python Files Configuring from Data Files Configuring from Environment Variables Configuration Best Practices Development Production Instance Folders Signals Core Signals Subscribing to Signals Creating Signals Sending Signals Signals and Flasks Request Context Decorator Based Signal Subscriptions Classbased Views Basic Reusable View URL Variables View Lifetime and self View Decorators Method Hints Method Dispatching and APIs Application Structure and Lifecycle Application Setup Serving the Application How a Request is Handled The Application Context Purpose of the Context Lifetime of the Context Manually Push a Context Storing Data Events and Signals The Request Context Purpose of the Context Lifetime of the Context Manually Push a Context How the Context Works Callbacks and Errors Notes On Proxies Modular Applications with Blueprints Why Blueprints The Concept of Blueprints My First Blueprint Registering Blueprints Nesting Blueprints Blueprint Resources Building URLs Blueprint Error Handlers Extensions Finding Extensions Using Extensions Building Extensions Command Line Interface Application Discovery Run the Development Server Open a Shell Environment Variables From dotenv Environment Variables From virtualenv Custom Commands Plugins Custom Scripts PyCharm Integration Development Server Command Line In Code Working with the Shell Command Line Interface Creating a Request Context Firing Before/After Request Further Improving the Shell Experience Patterns for Flask Large Applications as Packages Application Factories Application Dispatching Using URL Processors Using SQLite 3 with Flask SQLAlchemy in Flask Uploading Files Caching View Decorators Form Validation with WTForms Template Inheritance Message Flashing JavaScript, fetch, and JSON Lazily Loading Views MongoDB with MongoEngine Adding a favicon Streaming Contents Deferred Request Callbacks Adding HTTP Method Overrides Request Content Checksums Background Tasks with Celery Subclassing Flask SinglePage Applications Security

Considerations Resource Use CrossSite Scripting XSS CrossSite Request Forgery CSRF JSON Security Security Headers CopyPaste to Terminal Deploying to Production SelfHosted Options Hosting Platforms Using async and await Performance Background tasks When to use Quart instead Extensions Other event loops API Reference If you are looking for information on a specific function, class or method, this part of the documentation is for you.

Entities

Persons

- Flask SQLAlchemy

Organizations

- Celery Subclassing Flask
- MongoEngine
- Sphinx 8.2.3
- JSON
- Custom Commands Plugins Custom Scripts PyCharm Integration Development Server Command Line
- Application Structure and Lifecycle Application Setup Serving the Application How a Request
- the Shell Command Line Interface Creating a Request
- WSGI
- API Reference
- Models Recommended Extension Guidelines Contributing BSD3Clause License Changes Version
- Installation Python Version Dependencies Virtual
- Flask Flask Documentation
- Quickstart
- Access the Database Blueprints
- Werkzeug
- Flask Documentation
- Based Signal Subscriptions
- CrossSite
- API
- Flask
- View Decorators Method Hints Method Dispatching
- Terminal Deploying
- CopyPaste
- Installation
- 0.5.2 Version
- Views Templates Static Files Blog Blueprint Make the Project Installable Test Coverage Deploy
- Admins Injecting Request Information Other Libraries Configuration Handling Configuration Basics
- Thread Locals Asyncawait
- Performance Background
- Install Flask Quickstart A Minimal Application Debug Mode HTML Escaping Routing Static Files Rendering Templates Accessing Request Data Redirects
- WTFForms Template Inheritance Message Flashing JavaScript
- Flasks
- Flasks Request Context
- an Active Context Handling Application Errors
- Autoescaping Registering Filters
- WSGI Middleware Using Flask Extensions Deploying
- Production SelfHosted Options Hosting Platforms Using

- JSON Lazily Loading Views MongoDB

Locations

- Jinja