# Документация проекта

Без названия

## Flask

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. It began as a simple wrapper around [Werkzeug] and [Jinja], and …

Дата генерации: 10.05.2025 00:44

## Оглавление

## Древо проекта

```
    tmpbe5cunld/
        CHANGES.rst
        LICENSE.txt
        pyproject.toml
        README.md
        tox.ini
        docs/
            api.rst
            appcontext.rst
            async-await.rst
            blueprints.rst
            changes.rst
            cli.rst
            conf.py
            config.rst
            contributing.rst
            debugging.rst
            design.rst
            errorhandling.rst
            extensiondev.rst
            extensions.rst
```

```
            index.rst
            installation.rst
            license.rst
            lifecycle.rst
            logging.rst
            make.bat
            Makefile
            quickstart.rst
            reqcontext.rst
            server.rst
            shell.rst
            signals.rst
            templating.rst
            testing.rst
            views.rst
            web-security.rst
            deploying/
                apache-httpd.rst
                asgi.rst
                eventlet.rst
                gevent.rst
                gunicorn.rst
                index.rst
                mod_wsgi.rst
```

```
            nginx.rst
            proxy_fix.rst
            uwsgi.rst
            waitress.rst
        patterns/
            appdispatch.rst
            appfactories.rst
            caching.rst
            celery.rst
            deferredcallbacks.rst
            favicon.rst
            fileuploads.rst
            flashing.rst
            index.rst
            javascript.rst
            jquery.rst
            lazyloading.rst
            methodoverrides.rst
            mongoengine.rst
            packages.rst
            requestchecksum.rst
            singlepageapplications.rst
            sqlalchemy.rst
            sqlite3.rst
```

```
        streaming.rst
        subclassing.rst
        templateinheritance.rst
        urlprocessors.rst
        viewdecorators.rst
        wtforms.rst
    tutorial/
        blog.rst
        database.rst
        deploy.rst
        factory.rst
        flaskr_edit.png
        flaskr_index.png
        flaskr_login.png
        index.rst
        install.rst
        layout.rst
        next.rst
        static.rst
        templates.rst
        tests.rst
        views.rst
    _static/
        debugger.png
```

```
            flask-horizontal.png
            flask-vertical.png
            pycharm-run-config.png
            shortcut-icon.png
    examples/
        celery/
            make_celery.py
            pyproject.toml
            README.md
            requirements.txt
            src/
                task_app/
                    tasks.py
                    views.py
                    __init__.py
                    templates/
                        index.html
        javascript/
            LICENSE.txt
            pyproject.toml
            README.rst
            js_example/
                views.py
                __init__.py
```

```
                templates/
                    base.html
                    fetch.html
                    jquery.html
                    xhr.html
            tests/
                conftest.py
                test_js_example.py
        tutorial/
            LICENSE.txt
            pyproject.toml
            README.rst
            flaskr/
                auth.py
                blog.py
                db.py
                schema.sql
                __init__.py
                static/
                    style.css
                templates/
                    base.html
                    auth/
                        login.html
```

```
                        register.html
                    blog/
                        create.html
                        index.html
                        update.html
            tests/
                conftest.py
                data.sql
                test_auth.py
                test_blog.py
                test_db.py
                test_factory.py
    requirements/
        build.in
        build.txt
        dev.in
        dev.txt
        docs.in
        docs.txt
        tests-dev.txt
        tests-min.in
        tests-min.txt
        tests.in
        tests.txt
```

```
        typing.in
        typing.txt
    src/
        flask/
            app.py
            blueprints.py
            cli.py
            config.py
            ctx.py
            debughelpers.py
            globals.py
            helpers.py
            logging.py
            py.typed
            sessions.py
            signals.py
            templating.py
            testing.py
            typing.py
            views.py
            wrappers.py
            __init__.py
            __main__.py
            json/
```

```
                provider.py
                tag.py
                __init__.py
            sansio/
                app.py
                blueprints.py
                README.md
                scaffold.py
        tests/
            conftest.py
            test_appctx.py
            test_async.py
            test_basic.py
            test_blueprints.py
            test_cli.py
            test_config.py
            test_converters.py
            test_helpers.py
            test_instance_config.py
            test_json.py
            test_json_tag.py
            test_logging.py
            test_regression.py
            test_reqctx.py
```

```
            test_request.py
            test_session_interface.py
            test_signals.py
            test_subclassing.py
            test_templating.py
            test_testing.py
            test_user_error_handler.py
            test_views.py
            static/
                config.json
                config.toml
                index.html
            templates/
                context_template.html
                escaping_template.html
                mail.txt
                non_escaping_template.txt
                simple_template.html
                template_filter.html
                template_test.html
                _macro.html
                nested/
                    nested.txt
            test_apps/
```

```
blueprintapp/
    __init__.py
    apps/
        __init__.py
        admin/
            __init__.py
            static/
                test.txt
                css/
                    test.css
            templates/
                admin/
                    index.html
        frontend/
            __init__.py
            templates/
                frontend/
                    index.html
cliapp/
    app.py
    factory.py
    importerrorapp.py
    message.txt
    multiapp.py
```

```
                __init__.py
                inner1/
                    __init__.py
                    inner2/
                        flask.py
                        __init__.py
            helloworld/
                hello.py
                wsgi.py
            subdomaintestmodule/
                __init__.py
                static/
                    hello.txt
        type_check/
            typing_app_decorators.py
            typing_error_handler.py
            typing_route.py
```

## Введение

> # Flask
>
> 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. It

began as a simple wrapper around [Werkzeug] and [Jinja], and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

## A Simple Example

```python

# save this as app.py

from flask import Flask

app = Flask(**name**)

@app.route("/") def hello(): return "Hello, World!" ```

```
$ flask run * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

## Donate

The Pallets organization develops and supports Flask and the libraries it uses. In order to grow the community of contributors and users, and allow the maintainers to devote more time to the projects, [please donate today].

## Contributing

See our detailed contributing documentation for many ways to contribute, including reporting issues, requesting features, asking or answering questions, and making PRs.
```

# Исходные файлы

## docs\conf.py

**Функции:**

- **github_link**
- **setup**

## examples\celery\make_celery.py

## examples\celery\src\task_app\tasks.py

**Функции:**

- **add**
- **block**
- **process**

## examples\celery\src\task_app\views.py

**Функции:**

- **result**
- **add**
- **block**
- **process**

**API Endpoints:**

| Function | Decorator | Path |
|---|---|---|
| result | get | /result/<id> |
| add | post | /add |

| | | |
|---|---|---|
| block | post | /block |
| process | post | /process |

## examples\celery\src\task_app__init__.py

**Функции:**

- **create_app**
- **celery_init_app**

## examples\javascript\js_example\views.py

**Функции:**

- **index**
- **add**

**API Endpoints:**

| Function | Decorator | Path |
|---|---|---|
| index | route | / |
| index | route | /<any(xhr, jquery, fetch):js> |
| add | route | /add |

## examples\javascript\js_example__init__.py

## examples\javascript\tests\conftest.py

**Функции:**

- **fixture_app**
- **client**

## examples\javascript\tests\test_js_example.py

**Функции:**

- **test_index**
- **test_add**

## examples\tutorial\flaskr\auth.py

**Функции:**

- **login_required**
  *View decorator that redirects anonymous users to the login page.*
- **load_logged_in_user**
  *If a user id is stored in the session, load the user object from the database into ``g.user``.*
- **register**
  *Register a new user.*

  *Validates that the username is not already taken. Hashes the password for security.*
- **login**
  *Log in a registered user by adding the user id to the session.*
- **logout**

*Clear the current session, including the stored user id.*

**API Endpoints:**

| Function | Decorator | Path |
|---|---|---|
| register | route | /register |
| login | route | /login |
| logout | route | /logout |

## examples\tutorial\flaskr\blog.py

**Функции:**

- **index**
  *Show all the posts, most recent first.*
- **get_post**
  *Get a post and its author by id.*

  *Checks that the id exists and optionally that the current user is the author.*

  *:param id: id of post to get*
  *:param check_author: require the current user to be the author*
  *:return: the post with author information*
  *:raise 404: if a post with the given id doesn't exist*
  *:raise 403: if the current user isn't the author*
- **create**
  *Create a new post for the current user.*
- **update**
  *Update a post if the current user is the author.*
- **delete**
  *Delete a post.*

  *Ensures that the post exists and that the logged in user is the author of the post.*

**API Endpoints:**

| Function | Decorator | Path |
|---|---|---|
| index | route | / |

| | | |
|---|---|---|
| create | route | /create |
| update | route | /<int:id>/update |
| delete | route | /<int:id>/delete |

## examples\tutorial\flaskr\db.py

**Функции:**

- **get_db**
  *Connect to the application's configured database. The connection is unique for each request and will be reused if this is called again.*
- **close_db**
  *If this request connected to the database, close the connection.*
- **init_db**
  *Clear existing data and create new tables.*
- **init_db_command**
  *Clear existing data and create new tables.*
- **init_app**
  *Register database functions with the Flask app. This is called by the application factory.*

## examples\tutorial\flaskr__init__.py

**Функции:**

- **create_app**
  *Create and configure an instance of the Flask application.*

## examples\tutorial\tests\conftest.py

**Классы:**

- **AuthActions**

**Функции:**

- **app**
  *Create and configure a new app instance for each test.*
- **client**
  *A test client for the app.*
- **runner**
  *A test runner for the app's Click commands.*
- **auth**

## examples\tutorial\tests\test_auth.py

**Функции:**

- **test_register**
- **test_register_validate_input**
- **test_login**
- **test_login_validate_input**
- **test_logout**

## examples\tutorial\tests\test_blog.py

**Функции:**

- **test_index**
- **test_login_required**
- **test_author_required**
- **test_exists_required**
- **test_create**
- **test_update**
- **test_create_update_validate**
- **test_delete**

## examples\tutorial\tests\test_db.py

**Функции:**

- **test_get_close_db**
- **test_init_db_command**

## examples\tutorial\tests\test_factory.py

**Функции:**

- **test_config**
  *Test create_app without passing test config.*
- **test_hello**

## src\flask\app.py

**Классы:**

- **Flask**

*The flask object implements a WSGI application and acts as the central object.  It is passed the name of the module or package of the application.  Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.*

*The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an :file:`__init__.py` file inside) or a standard module (just a ``.py`` file).*

*For more information about resource loading, see :func:`open_resource`.*

*Usually you create a :class:`Flask` instance in your main module or in the :file:`__init__.py` file of your package like this::*

```
from flask import Flask
app = Flask(__name__)
```

*.. admonition:: About the First Parameter*

*The idea of the first parameter is to give Flask an idea of what belongs to your application.  This name is used to find resources on the filesystem, can be used by extensions to improve debugging information and a lot more.*

*So it's important what you provide there.  If you are using a single module, `__name__` is always the correct value.  If you however are using a package, it's usually recommended to hardcode the name of your package there.*

*For example if your application is defined in :file:`yourapplication/app.py` you should create it with one of the two versions below::*

```
app = Flask('yourapplication')
app = Flask(__name__.split('.')[0])
```

*Why is that?  The application will work even with `__name__`, thanks to how resources are looked up.  However it will make debugging more painful.  Certain extensions can make assumptions based on the import name of your application.  For example the Flask-SQLAlchemy extension will look for the code in your application that triggered an SQL query in debug mode.  If the import name is not properly set up, that debugging information is lost.  (For example it would only pick up SQL queries in `yourapplication.app` and not `yourapplication.views.frontend`)*

*.. versionadded:: 0.7*
*The `static_url_path`, `static_folder`, and `template_folder` parameters were added.*

*.. versionadded:: 0.8*
*The `instance_path` and `instance_relative_config` parameters were added.*

*.. versionadded:: 0.11*
*The `root_path` parameter was added.*

*.. versionadded:: 1.0*
*The ``host_matching`` and ``static_host`` parameters were added.*

*.. versionadded:: 1.0*
*The ``subdomain_matching`` parameter was added. Subdomain matching needs to be enabled manually now. Setting :data:`SERVER_NAME` does not implicitly enable it.*

*:param import_name: the name of the application package*
*:param static_url_path: can be used to specify a different path for the static files on the web. Defaults to the name of the `static_folder` folder.*
*:param static_folder: The folder with static files that is served at ``static_url_path``. Relative to the application ``root_path`` or an absolute path. Defaults to ``'static'``.*
*:param static_host: the host to use when adding the static route. Defaults to None. Required when using ``host_matching=True`` with a ``static_folder`` configured.*
*:param host_matching: set ``url_map.host_matching`` attribute. Defaults to False.*
*:param subdomain_matching: consider the subdomain relative to :data:`SERVER_NAME` when matching routes. Defaults to False.*
*:param template_folder: the folder that contains the templates that should be used by the application. Defaults to ``'templates'`` folder in the root path of the application.*
*:param instance_path: An alternative instance path for the application. By default the folder ``'instance'`` next to the package or module is assumed to be the instance path.*
*:param instance_relative_config: if set to ``True`` relative filenames for loading the config are assumed to be relative to the instance path instead of the application root.*
*:param root_path: The path to the root of the application files. This should only be set manually when it can't be detected automatically, such as for namespace packages.*

**Функции:**

- **_make_timedelta**

## src\flask\blueprints.py

**Классы:**

- **Blueprint**

## src\flask\cli.py

**Классы:**

- **NoAppException**
  *Raised if an application cannot be found or loaded.*
- **ScriptInfo**
  *Helper object to deal with Flask applications.  This is usually not necessary to interface with as it's used internally in the dispatching to click.  In future versions of Flask this object will most likely play a bigger role.  Typically it's created automatically by the :class:`FlaskGroup` but you can also manually create it and pass it onwards as click object.*

  *.. versionchanged:: 3.1*
  *Added the ``load_dotenv_defaults`` parameter and attribute.*
- **AppGroup**
  *This works similar to a regular click :class:`~click.Group` but it changes the behavior of the :meth:`command` decorator so that it automatically wraps the functions in :func:`with_appcontext`.*

  *Not to be confused with :class:`FlaskGroup`.*
- **FlaskGroup**
  *Special subclass of the :class:`AppGroup` group that supports loading more commands from the configured Flask app.  Normally a developer does not have to interface with this class but there are some very advanced use cases for which it makes sense to create an instance of this. see :ref:`custom-scripts`.*

  *:param add_default_commands: if this is True then the default run and*
  *shell commands will be added.*
  *:param add_version_option: adds the ``--version`` option.*
  *:param create_app: an optional callback that is passed the script info and*
  *returns the loaded app.*
  *:param load_dotenv: Load the nearest :file:`.env` and :file:`.flaskenv`*

*files to set environment variables. Will also change the working directory to the directory containing the first file found.*
*:param set_debug_flag: Set the app's debug flag.*

*.. versionchanged:: 3.1*
*``-e path`` takes precedence over default ``.env`` and ``.flaskenv`` files.*

*.. versionchanged:: 2.2*
*Added the ``-A/--app``, ``--debug/--no-debug``, ``-e/--env-file`` options.*

*.. versionchanged:: 2.2*
*An app context is pushed when running ``app.cli`` commands, so ``@with_appcontext`` is no longer required for those commands.*

*.. versionchanged:: 1.0*
*If installed, python-dotenv will be used to load environment variables from :file:`.env` and :file:`.flaskenv` files.*

- **CertParamType**
*Click option type for the ``--cert`` option. Allows either an existing file, the string ``'adhoc'``, or an import for a :class:`~ssl.SSLContext` object.*
- **SeparatedPathType**
*Click option type that accepts a list of values separated by the OS's path separator (``:``, ``;`` on Windows). Each value is validated as a :class:`click.Path` type.*

**Функции:**

- **find_best_app**
*Given a module instance this tries to find the best possible application in the module or raises an exception.*
- **_called_with_wrong_args**
*Check whether calling a function raised a ``TypeError`` because the call failed or because something in the factory raised the error.*

*:param f: The function that was called.*
*:return: ``True`` if the call failed.*
- **find_app_by_string**
*Check if the given string is a variable name or a function. Call a function to get the app instance, or return the variable directly.*
- **prepare_import**
*Given a filename this will try to calculate the python path, add it to the search path and return the actual module name that is expected.*
- **locate_app**
- **locate_app**

- **locate_app**
- **get_version**
- **with_appcontext**
  *Wraps a callback so that it's guaranteed to be executed with the script's application context.*

  *Custom commands (and their options) registered under ``app.cli`` or ``blueprint.cli`` will always have an app context available, this decorator is not required in that case.*

  *.. versionchanged:: 2.2*
  *The app context is active for subcommands as well as the decorated callback. The app context is always available to ``app.cli`` command and parameter callbacks.*
- **_set_app**
- **_set_debug**
- **_env_file_callback**
- **_path_is_ancestor**
  *Take ``other`` and remove the length of ``path`` from it. Then join it to ``path``. If it is the original value, ``path`` is an ancestor of ``other``.*
- **load_dotenv**
  *Load "dotenv" files to set environment variables. A given path takes precedence over ``.env``, which takes precedence over ``.flaskenv``. After loading and combining these files, values are only set if the key is not already set in ``os.environ``.*

  *This is a no-op if `python-dotenv`_ is not installed.*

  *.. _python-dotenv: https://github.com/theskumar/python-dotenv#readme*

  *:param path: Load the file at this location.*
  *:param load_defaults: Search for and load the default ``.flaskenv`` and ``.env`` files.*
  *:return: ``True`` if at least one env var was loaded.*

  *.. versionchanged:: 3.1*
  *Added the ``load_defaults`` parameter. A given path takes precedence over default files.*

  *.. versionchanged:: 2.0*
  *The current directory is not changed to the location of the loaded file.*

  *.. versionchanged:: 2.0*
  *When loading the env files, set the default encoding to UTF-8.*

  *.. versionchanged:: 1.1.0*

*Returns ``False`` when python-dotenv is not installed, or when the given path isn't a file.*

*.. versionadded:: 1.0*

- **show_server_banner**
  *Show extra startup messages the first time the server is run, ignoring the reloader.*
- **_validate_key**
  *The ``--key`` option must be specified when ``--cert`` is a file. Modifies the ``cert`` param to be a ``(cert, key)`` pair if needed.*
- **run_command**
  *Run a local development server.*

  *This server is for development purposes only. It does not provide the stability, security, or performance of production WSGI servers.*

  *The reloader and debugger are enabled by default with the '--debug' option.*
- **shell_command**
  *Run an interactive Python shell in the context of a given Flask application. The application will populate the default namespace of this shell according to its configuration.*

  *This is useful for executing small snippets of management code without having to manually configure the application.*
- **routes_command**
  *Show all registered routes with endpoints and methods.*
- **main**

## src\flask\config.py

**Классы:**

- **ConfigAttribute**
  *Makes an attribute forward to the config*
- **Config**
  *Works exactly like a dict but provides ways to fill it from files or special dictionaries. There are two common patterns to populate the config.*

  *Either you can fill the config from a config file::*

*app.config.from_pyfile('yourconfig.cfg')*

*Or alternatively you can define the configuration options in the module that calls :meth:`from_object` or provide an import path to a module that should be loaded. It is also possible to tell it to use the same module and with that provide the configuration values just before the call::*

*DEBUG = True*
*SECRET_KEY = 'development key'*
*app.config.from_object(__name__)*

*In both cases (loading from any Python file or loading from modules), only uppercase keys are added to the config. This makes it possible to use lowercase values in the config file for temporary values that are not added to the config or to define the config keys in the same file that implements the application.*

*Probably the most interesting way to load configurations is from an environment variable pointing to a file::*

*app.config.from_envvar('YOURAPPLICATION_SETTINGS')*

*In this case before launching the application you have to set this environment variable to the file you want to use. On Linux and OS X use the export statement::*

*export YOURAPPLICATION_SETTINGS='/path/to/config/file'*

*On windows use `set` instead.*

*:param root_path: path to which files are read relative from. When the config object is created by the application, this is the application's :attr:`~flask.Flask.root_path`.*
*:param defaults: an optional dictionary of default values*

## src\flask\ctx.py

**Классы:**

- **_AppCtxGlobals**
  *A plain object. Used as a namespace for storing data during an application context.*

  *Creating an app context automatically creates this object, which is*

*made available as the :data:`g` proxy.*

*.. describe:: 'key' in g*

*Check whether an attribute is present.*

*.. versionadded:: 0.10*

*.. describe:: iter(g)*

*Return an iterator over the attribute names.*

*.. versionadded:: 0.10*

- **AppContext**
  *The app context contains application-specific information. An app context is created and pushed at the beginning of each request if one is not already active. An app context is also pushed when running CLI commands.*
- **RequestContext**
  *The request context contains per-request information. The Flask app creates and pushes it at the beginning of the request, then pops it at the end of the request. It will create the URL adapter and request object for the WSGI environment provided.*

  *Do not attempt to use this class directly, instead use :meth:`~flask.Flask.test_request_context` and :meth:`~flask.Flask.request_context` to create this object.*

  *When the request context is popped, it will evaluate all the functions registered on the application for teardown execution (:meth:`~flask.Flask.teardown_request`).*

  *The request context is automatically popped at the end of the request. When using the interactive debugger, the context will be restored so ``request`` is still accessible. Similarly, the test client can preserve the context after the request ends. However, teardown functions may already have closed some resources such as database connections.*

**Функции:**

- **after_this_request**
  *Executes a function after this request.  This is useful to modify response objects.  The function is passed the response object and has to return the same or a new one.*

  *Example::*

*@app.route('/')*
*def index():*
*@after_this_request*
*def add_header(response):*
*response.headers['X-Foo'] = 'Parachute'*
*return response*
*return 'Hello World!'*

*This is more useful if a function other than the view function wants to modify a response. For instance think of a decorator that wants to add some headers without converting the return value into a response object.*

*.. versionadded:: 0.9*

- **copy_current_request_context**
*A helper function that decorates a function to retain the current request context. This is useful when working with greenlets. The moment the function is decorated a copy of the request context is created and then pushed when the function is called. The current session is also included in the copied request context.*

*Example::*

*import gevent*
*from flask import copy_current_request_context*

*@app.route('/')*
*def index():*
*@copy_current_request_context*
*def do_some_work():*
*# do some work here, it can access flask.request or*
*# flask.session like you would otherwise in the view function.*
*...*
*gevent.spawn(do_some_work)*
*return 'Regular response'*

*.. versionadded:: 0.10*

- **has_request_context**
*If you have code that wants to test if a request context is there or not this function can be used. For instance, you may want to take advantage*
*of request information if the request object is available, but fail silently if it is unavailable.*

*::*

*class User(db.Model):*

*def __init__(self, username, remote_addr=None):*

*self.username = username*
*if remote_addr is None and has_request_context():*
*remote_addr = request.remote_addr*
*self.remote_addr = remote_addr*

*Alternatively you can also just test any of the context bound objects (such as :class:`request` or :class:`g`) for truthness::*

*class User(db.Model):*

*def __init__(self, username, remote_addr=None):*
*self.username = username*
*if remote_addr is None and request:*
*remote_addr = request.remote_addr*
*self.remote_addr = remote_addr*

*.. versionadded:: 0.7*

- **has_app_context**
  *Works like :func:`has_request_context` but for the application context.  You can also just do a boolean check on the :data:`current_app` object instead.*

  *.. versionadded:: 0.9*

## src\flask\debughelpers.py

**Классы:**

- **UnexpectedUnicodeError**
  *Raised in places where we want some better error reporting for unexpected unicode or binary data.*
- **DebugFilesKeyError**
  *Raised from request.files during debugging.  The idea is that it can provide a better error message than just a generic KeyError/BadRequest.*
- **FormDataRoutingRedirect**
  *This exception is raised in debug mode if a routing redirect would cause the browser to drop the method or body. This happens when method is not GET, HEAD or OPTIONS and the status code is not 307 or 308.*

**Функции:**

- **attach_enctype_error_multidict**

*Patch ``request.files.__getitem__`` to raise a descriptive error about ``enctype=multipart/form-data``.*

*:param request: The request to patch.*
*:meta private:*

- **_dump_loader_info**
- **explain_template_loading_attempts**
  *This should help developers understand what failed*

## src\flask\globals.py

## src\flask\helpers.py

**Функции:**

- **get_debug_flag**
  *Get whether debug mode should be enabled for the app, indicated by the :envvar:`FLASK_DEBUG` environment variable. The default is ``False``.*
- **get_load_dotenv**
  *Get whether the user has disabled loading default dotenv files by setting :envvar:`FLASK_SKIP_DOTENV`. The default is ``True``, load the files.*

  *:param default: What to return if the env var isn't set.*
- **stream_with_context**
- **stream_with_context**
- **stream_with_context**
  *Request contexts disappear when the response is started on the server. This is done for efficiency reasons and to make it less likely to encounter memory leaks with badly written WSGI middlewares. The downside is that if*
  *you are using streamed responses, the generator cannot access request bound*
  *information any more.*

  *This function however can help you keep the context around for longer::*

  ```
  from flask import stream_with_context, request, Response
  ```

*@app.route('/stream')*
*def streamed_response():*
*@stream_with_context*
*def generate():*
*yield 'Hello '*
*yield request.args['name']*
*yield '!'*
*return Response(generate())*

*Alternatively it can also be used around a specific generator::*

*from flask import stream_with_context, request, Response*

*@app.route('/stream')*
*def streamed_response():*
*def generate():*
*yield 'Hello '*
*yield request.args['name']*
*yield '!'*
*return Response(stream_with_context(generate()))*

*.. versionadded:: 0.9*

- **make_response**
*Sometimes it is necessary to set additional headers in a view. Because views do not have to return response objects but can return a value that is converted into a response object by Flask itself, it becomes tricky to add headers to it. This function can be called instead of using a return and you will get a response object which you can use to attach headers.*

*If view looked like this and you want to add a new header::*

*def index():*
*return render_template('index.html', foo=42)*

*You can now do something like this::*

*def index():*
*response = make_response(render_template('index.html', foo=42))*
*response.headers['X-Parachutes'] = 'parachutes are cool'*
*return response*

*This function accepts the very same arguments you can return from a view function. This for example creates a response with a 404 error code::*

*response = make_response(render_template('not_found.html'), 404)*

*The other use case of this function is to force the return value of a*

*view function into a response which is helpful with view decorators::*

*response = make_response(view_function())*
*response.headers['X-Parachutes'] = 'parachutes are cool'*

*Internally this function does the following things:*

- *if no arguments are passed, it creates a new response argument*
- *if one argument is passed, :meth:`flask.Flask.make_response` is invoked with it.*
- *if more than one argument is passed, the arguments are passed to the :meth:`flask.Flask.make_response` function as tuple.*

*.. versionadded:: 0.6*

- **url_for**

*Generate a URL to the given endpoint with the given values.*

*This requires an active request or application context, and calls :meth:`current_app.url_for() <flask.Flask.url_for>`. See that method for full documentation.*

*:param endpoint: The endpoint name associated with the URL to generate. If this starts with a ``.``, the current blueprint name (if any) will be used.*
*:param _anchor: If given, append this as ``#anchor`` to the URL.*
*:param _method: If given, generate the URL associated with this method for the endpoint.*
*:param _scheme: If given, the URL will have this scheme if it is external.*
*:param _external: If given, prefer the URL to be internal (False) or require it to be external (True). External URLs include the scheme and domain. When not in an active request, URLs are external by default.*
*:param values: Values to use for the variable parts of the URL rule. Unknown keys are appended as query string arguments, like ``?a=b&c=d``.*

*.. versionchanged:: 2.2*
*Calls ``current_app.url_for``, allowing an app to override the behavior.*

*.. versionchanged:: 0.10*
*The ``_scheme`` parameter was added.*

*.. versionchanged:: 0.9*
*The ``_anchor`` and ``_method`` parameters were added.*

*.. versionchanged:: 0.9*
*Calls ``app.handle_url_build_error`` on build errors.*

- **redirect**
*Create a redirect response object.*

*If :data:`~flask.current_app` is available, it will use its*
*:meth:`~flask.Flask.redirect` method, otherwise it will use*
*:func:`werkzeug.utils.redirect`.*

*:param location: The URL to redirect to.*
*:param code: The status code for the redirect.*
*:param Response: The response class to use. Not used when*
*``current_app`` is active, which uses ``app.response_class``.*

*.. versionadded:: 2.2*
*Calls ``current_app.redirect`` if available instead of always*
*using Werkzeug's default ``redirect``.*

- **abort**
*Raise an :exc:`~werkzeug.exceptions.HTTPException` for the given*
*status code.*

*If :data:`~flask.current_app` is available, it will call its*
*:attr:`~flask.Flask.aborter` object, otherwise it will use*
*:func:`werkzeug.exceptions.abort`.*

*:param code: The status code for the exception, which must be*
*registered in ``app.aborter``.*
*:param args: Passed to the exception.*
*:param kwargs: Passed to the exception.*

*.. versionadded:: 2.2*
*Calls ``current_app.aborter`` if available instead of always*
*using Werkzeug's default ``abort``.*

- **get_template_attribute**
*Loads a macro (or variable) a template exports.  This can be used to*
*invoke a macro from within Python code.  If you for example have a*
*template named :file:`_cider.html` with the following contents:*

*.. sourcecode:: html+jinja*

*{% macro hello(name) %}Hello {{ name }}!{% endmacro %}*

*You can access this from Python code like this::*

*hello = get_template_attribute('_cider.html', 'hello')*
*return hello('World')*

*.. versionadded:: 0.2*

*:param template_name: the name of the template*
*:param attribute: the name of the variable of macro to access*

- **flash**
  *Flashes a message to the next request.  In order to remove the*
  *flashed message from the session and to display it to the user,*
  *the template has to call :func:`get_flashed_messages`.*

  *.. versionchanged:: 0.3*
  *`category` parameter added.*

  *:param message: the message to be flashed.*
  *:param category: the category for the message.  The following values*
  *are recommended: ``'message'`` for any kind of message,*
  *``'error'`` for errors, ``'info'`` for information*
  *messages and ``'warning'`` for warnings.  However any*
  *kind of string can be used as category.*
- **get_flashed_messages**
  *Pulls all flashed messages from the session and returns them.*
  *Further calls in the same request to the function will return*
  *the same messages.  By default just the messages are returned,*
  *but when `with_categories` is set to ``True``, the return value will*
  *be a list of tuples in the form ``(category, message)`` instead.*

  *Filter the flashed messages to one or more categories by providing those*
  *categories in `category_filter`.  This allows rendering categories in*
  *separate html blocks.  The `with_categories` and `category_filter`*
  *arguments are distinct:*

  ** `with_categories` controls whether categories are returned with message*
  *text (``True`` gives a tuple, where ``False`` gives just the message text).*
  ** `category_filter` filters the messages down to only those matching the*
  *provided categories.*

  *See :doc:`/patterns/flashing` for examples.*

  *.. versionchanged:: 0.3*
  *`with_categories` parameter added.*

  *.. versionchanged:: 0.9*
  *`category_filter` parameter added.*

  *:param with_categories: set to ``True`` to also receive categories.*
  *:param category_filter: filter of categories to limit return values.  Only*
  *categories in the list will be returned.*
- **_prepare_send_file_kwargs**

- **send_file**
  *Send the contents of a file to the client.*

  *The first argument can be a file path or a file-like object. Paths are preferred in most cases because Werkzeug can manage the file and get extra information from the path. Passing a file-like object requires that the file is opened in binary mode, and is mostly useful when building a file in memory with :class:`io.BytesIO`.*

  *Never pass file paths provided by a user. The path is assumed to be trusted, so a user could craft a path to access a file you didn't intend. Use :func:`send_from_directory` to safely serve user-requested paths from within a directory.*

  *If the WSGI server sets a ``file_wrapper`` in ``environ``, it is used, otherwise Werkzeug's built-in wrapper is used. Alternatively, if the HTTP server supports ``X-Sendfile``, configuring Flask with ``USE_X_SENDFILE = True`` will tell the server to send the given path, which is much more efficient than reading it in Python.*

  *:param path_or_file: The path to the file to send, relative to the current working directory if a relative path is given. Alternatively, a file-like object opened in binary mode. Make sure the file pointer is seeked to the start of the data.*
  *:param mimetype: The MIME type to send for the file. If not provided, it will try to detect it from the file name.*
  *:param as_attachment: Indicate to a browser that it should offer to save the file instead of displaying it.*
  *:param download_name: The default name browsers will use when saving the file. Defaults to the passed file name.*
  *:param conditional: Enable conditional and range responses based on request headers. Requires passing a file path and ``environ``.*
  *:param etag: Calculate an ETag for the file, which requires passing a file path. Can also be a string to use instead.*
  *:param last_modified: The last modified time to send for the file, in seconds. If not provided, it will try to detect it from the file path.*
  *:param max_age: How long the client should cache the file, in seconds. If set, ``Cache-Control`` will be ``public``, otherwise it will be ``no-cache`` to prefer conditional caching.*

  *.. versionchanged:: 2.0*
  *``download_name`` replaces the ``attachment_filename`` parameter. If ``as_attachment=False``, it is passed with ``Content-Disposition: inline`` instead.*

  *.. versionchanged:: 2.0*
  *``max_age`` replaces the ``cache_timeout`` parameter. ``conditional`` is enabled and ``max_age`` is not set by*

*default.*

*.. versionchanged:: 2.0*
*``etag`` replaces the ``add_etags`` parameter. It can be a string to use instead of generating one.*

*.. versionchanged:: 2.0*
*Passing a file-like object that inherits from :class:`~io.TextIOBase` will raise a :exc:`ValueError` rather than sending an empty file.*

*.. versionadded:: 2.0*
*Moved the implementation to Werkzeug. This is now a wrapper to pass some Flask-specific arguments.*

*.. versionchanged:: 1.1*
*``filename`` may be a :class:`~os.PathLike` object.*

*.. versionchanged:: 1.1*
*Passing a :class:`~io.BytesIO` object supports range requests.*

*.. versionchanged:: 1.0.3*
*Filenames are encoded with ASCII instead of Latin-1 for broader compatibility with WSGI servers.*

*.. versionchanged:: 1.0*
*UTF-8 filenames as specified in :rfc:`2231` are supported.*

*.. versionchanged:: 0.12*
*The filename is no longer automatically inferred from file objects. If you want to use automatic MIME and etag support, pass a filename via ``filename_or_fp`` or ``attachment_filename``.*

*.. versionchanged:: 0.12*
*``attachment_filename`` is preferred over ``filename`` for MIME detection.*

*.. versionchanged:: 0.9*
*``cache_timeout`` defaults to :meth:`Flask.get_send_file_max_age`.*

*.. versionchanged:: 0.7*
*MIME guessing and etag support for file-like objects was removed because it was unreliable. Pass a filename if you are able to, otherwise attach an etag yourself.*

*.. versionchanged:: 0.5*
*The ``add_etags``, ``cache_timeout`` and ``conditional``*

*parameters were added. The default behavior is to add etags.*

*.. versionadded:: 0.2*

- **send_from_directory**
  *Send a file from within a directory using :func:`send_file`.*

  *.. code-block:: python*

  *@app.route("/uploads/<path:name>")*
  *def download_file(name):*
  *return send_from_directory(*
  *app.config['UPLOAD_FOLDER'], name, as_attachment=True*
  *)*

  *This is a secure way to serve files from a folder, such as static files or uploads. Uses :func:`~werkzeug.security.safe_join` to ensure the path coming from the client is not maliciously crafted to point outside the specified directory.*

  *If the final path does not point to an existing regular file, raises a 404 :exc:`~werkzeug.exceptions.NotFound` error.*

  *:param directory: The directory that ``path`` must be located under, relative to the current application's root path. This *must not* be a value provided by the client, otherwise it becomes insecure.*
  *:param path: The path to the file to send, relative to ``directory``.*
  *:param kwargs: Arguments to pass to :func:`send_file`.*

  *.. versionchanged:: 2.0*
  *``path`` replaces the ``filename`` parameter.*

  *.. versionadded:: 2.0*
  *Moved the implementation to Werkzeug. This is now a wrapper to pass some Flask-specific arguments.*

  *.. versionadded:: 0.5*

- **get_root_path**
  *Find the root path of a package, or the path that contains a module. If it cannot be found, returns the current working directory.*

  *Not to be confused with the value returned by :func:`find_package`.*

  *:meta private:*

- **_split_blueprint_path**

## src\flask\logging.py

**Функции:**

- **wsgi_errors_stream**
  *Find the most appropriate error stream for the application. If a request is active, log to ``wsgi.errors``, otherwise use ``sys.stderr``.*

  *If you configure your own :class:`logging.StreamHandler`, you may want to use this for the stream. If you are using file or dict configuration and can't import this directly, you can refer to it as ``ext://flask.logging.wsgi_errors_stream``.*
- **has_level_handler**
  *Check if there is a handler in the logging chain that will handle the given logger's :meth:`effective level <~logging.Logger.getEffectiveLevel>`.*
- **create_logger**
  *Get the Flask app's logger and configure it if needed.*

  *The logger name will be the same as :attr:`app.import_name <flask.Flask.name>`.*

  *When :attr:`~flask.Flask.debug` is enabled, set the logger level to :data:`logging.DEBUG` if it is not set.*

  *If there is no handler for the logger's effective level, add a :class:`~logging.StreamHandler` for :func:`~flask.logging.wsgi_errors_stream` with a basic format.*

## src\flask\sessions.py

**Классы:**

- **SessionMixin**
  *Expands a basic dictionary with session attributes.*
- **SecureCookieSession**
  *Base class for sessions based on signed cookies.*

  *This session backend will set the :attr:`modified` and :attr:`accessed` attributes. It cannot reliably track whether a session is new (vs. empty), so :attr:`new` remains hard coded to ``False``.*

- **NullSession**
  *Class used to generate nicer error messages if sessions are not available.  Will still allow read-only access to the empty session but fail on setting.*
- **SessionInterface**
  *The basic interface you have to implement in order to replace the default session interface which uses werkzeug's securecookie implementation.  The only methods you have to implement are :meth:`open_session` and :meth:`save_session`, the others have useful defaults which you don't need to change.*

  *The session object returned by the :meth:`open_session` method has to provide a dictionary like interface plus the properties and methods from the :class:`SessionMixin`.  We recommend just subclassing a dict and adding that mixin::*

  ```
  class Session(dict, SessionMixin):
      pass
  ```

  *If :meth:`open_session` returns ``None`` Flask will call into :meth:`make_null_session` to create a session that acts as replacement if the session support cannot work because some requirement is not fulfilled.  The default :class:`NullSession` class that is created will complain that the secret key was not set.*

  *To replace the session interface on an application all you have to do is to assign :attr:`flask.Flask.session_interface`::*

  ```
  app = Flask(__name__)
  app.session_interface = MySessionInterface()
  ```

  *Multiple requests with the same session may be sent and handled concurrently. When implementing a new session interface, consider whether reads or writes to the backing store must be synchronized. There is no guarantee on the order in which the session for each request is opened or saved, it will occur in the order that requests begin and end processing.*

  *.. versionadded:: 0.8*
- **SecureCookieSessionInterface**
  *The default session interface that stores sessions in signed cookies through the :mod:`itsdangerous` module.*

**Функции:**

- **_lazy_sha1**

*Don't access ``hashlib.sha1`` until runtime. FIPS builds may not include SHA-1, in which case the import and use as a default would fail before the developer can configure something else.*

## src\flask\signals.py

## src\flask\templating.py

**Классы:**

- **Environment**
  *Works like a regular Jinja2 environment but has some additional knowledge of how Flask's blueprint works so that it can prepend the name of the blueprint to referenced templates if necessary.*
- **DispatchingJinjaLoader**
  *A loader that looks for templates in the application and all the blueprint folders.*

**Функции:**

- **_default_template_ctx_processor**
  *Default template context processor.  Injects `request`, `session` and `g`.*
- **_render**
- **render_template**
  *Render a template by name with the given context.*

  *:param template_name_or_list: The name of the template to render. If a list is given, the first name to exist will be rendered.*
  *:param context: The variables to make available in the template.*
- **render_template_string**
  *Render a template from the given source string with the given context.*

  *:param source: The source code of the template to render.*
  *:param context: The variables to make available in the template.*
- **_stream**
- **stream_template**

*Render a template by name with the given context as a stream.*
*This returns an iterator of strings, which can be used as a*
*streaming response from a view.*

*:param template_name_or_list: The name of the template to render. If*
*a list is given, the first name to exist will be rendered.*
*:param context: The variables to make available in the template.*

*.. versionadded:: 2.2*

- **stream_template_string**
  *Render a template from the given source string with the given*
  *context as a stream. This returns an iterator of strings, which can*
  *be used as a streaming response from a view.*

  *:param source: The source code of the template to render.*
  *:param context: The variables to make available in the template.*

  *.. versionadded:: 2.2*

## src\flask\testing.py

**Классы:**

- **EnvironBuilder**
  *An :class:`~werkzeug.test.EnvironBuilder`, that takes defaults from the*
  *application.*

  *:param app: The Flask application to configure the environment from.*
  *:param path: URL path being requested.*
  *:param base_url: Base URL where the app is being served, which*
  *``path`` is relative to. If not given, built from*
  *:data:`PREFERRED_URL_SCHEME`, ``subdomain``,*
  *:data:`SERVER_NAME`, and :data:`APPLICATION_ROOT`.*
  *:param subdomain: Subdomain name to append*
  *to :data:`SERVER_NAME`.*
  *:param url_scheme: Scheme to use instead of*
  *:data:`PREFERRED_URL_SCHEME`.*
  *:param json: If given, this is serialized as JSON and passed as*
  *``data``. Also defaults ``content_type`` to*
  *``application/json``.*
  *:param args: other positional arguments passed to*
  *:class:`~werkzeug.test.EnvironBuilder`.*
  *:param kwargs: other keyword arguments passed to*
  *:class:`~werkzeug.test.EnvironBuilder`.*

- **FlaskClient**
  *Works like a regular Werkzeug test client but has knowledge about Flask's contexts to defer the cleanup of the request context until the end of a ``with`` block. For general information about how to use this class refer to :class:`werkzeug.test.Client`.*

  *.. versionchanged:: 0.12*
  *`app.test_client()` includes preset default environment, which can be set after instantiation of the `app.test_client()` object in `client.environ_base`.*

  *Basic usage is outlined in the :doc:`/testing` chapter.*
- **FlaskCliRunner**
  *A :class:`~click.testing.CliRunner` for testing a Flask app's CLI commands. Typically created using :meth:`~flask.Flask.test_cli_runner`. See :ref:`testing-cli`.*

**Функции:**

- **_get_werkzeug_version**

## src\flask\typing.py

## src\flask\views.py

**Классы:**

- **View**
  *Subclass this class and override :meth:`dispatch_request` to create a generic class-based view. Call :meth:`as_view` to create a view function that creates an instance of the class with the given arguments and calls its ``dispatch_request`` method with any URL variables.*

  *See :doc:`views` for a detailed guide.*

  *.. code-block:: python*

  ```
  class Hello(View):
      init_every_request = False
  ```

*def dispatch_request(self, name):*
*return f"Hello, {name}!"*

*app.add_url_rule(*
*"/hello/<name>", view_func=Hello.as_view("hello")*
*)*

*Set :attr:`methods` on the class to change what methods the view accepts.*

*Set :attr:`decorators` on the class to apply a list of decorators to the generated view function. Decorators applied to the class itself will not be applied to the generated view function!*

*Set :attr:`init_every_request` to ``False`` for efficiency, unless you need to store request-global data on ``self``.*

- **MethodView**
  *Dispatches request methods to the corresponding instance methods. For example, if you implement a ``get`` method, it will be used to handle ``GET`` requests.*

  *This can be useful for defining a REST API.*

  *:attr:`methods` is automatically set based on the methods defined on the class.*

  *See :doc:`views` for a detailed guide.*

  *.. code-block:: python*

  *class CounterAPI(MethodView):*
  *def get(self):*
  *return str(session.get("counter", 0))*

  *def post(self):*
  *session["counter"] = session.get("counter", 0) + 1*
  *return redirect(url_for("counter"))*

  *app.add_url_rule(*
  *"/counter", view_func=CounterAPI.as_view("counter")*
  *)*

## src\flask\wrappers.py

**Классы:**

- **Request**
  *The request object used by default in Flask.  Remembers the matched endpoint and view arguments.*

  *It is what ends up as :class:`~flask.request`.  If you want to replace the request object used you can subclass this and set :attr:`~flask.Flask.request_class` to your subclass.*

  *The request object is a :class:`~werkzeug.wrappers.Request` subclass and provides all of the attributes Werkzeug defines plus a few Flask specific ones.*
- **Response**
  *The response object that is used by default in Flask.  Works like the response object from Werkzeug but is set to have an HTML mimetype by default.  Quite often you don't have to create this object yourself because :meth:`~flask.Flask.make_response` will take care of that for you.*

  *If you want to replace the response object used you can subclass this and set :attr:`~flask.Flask.response_class` to your subclass.*

  *.. versionchanged:: 1.0*
  *JSON support is added to the response, like the request. This is useful when testing to get the test client response data as JSON.*

  *.. versionchanged:: 1.0*

  *Added :attr:`max_cookie_size`.*

## src\flask__init__.py

## src\flask__main__.py

## src\flask\json\provider.py

**Классы:**

- **JSONProvider**
  *A standard set of JSON operations for an application. Subclasses of this can be used to customize JSON behavior or use different JSON libraries.*

  *To implement a provider for a specific library, subclass this base class and implement at least :meth:`dumps` and :meth:`loads`. All other methods have default implementations.*

  *To use a different provider, either subclass ``Flask`` and set :attr:`~flask.Flask.json_provider_class` to a provider class, or set :attr:`app.json <flask.Flask.json>` to an instance of the class.*

  *:param app: An application instance. This will be stored as a :class:`weakref.proxy` on the :attr:`_app` attribute.*

  *.. versionadded:: 2.2*
- **DefaultJSONProvider**
  *Provide JSON operations using Python's built-in :mod:`json` library. Serializes the following additional data types:*

  - *:class:`datetime.datetime` and :class:`datetime.date` are serialized to :rfc:`822` strings. This is the same as the HTTP date format.*
  - *:class:`uuid.UUID` is serialized to a string.*
  - *:class:`dataclasses.dataclass` is passed to :func:`dataclasses.asdict`.*
  - *:class:`~markupsafe.Markup` (or any object with a ``__html__`` method) will call the ``__html__`` method to get a string.*

**Функции:**

- **_default**

## src\flask\json\tag.py

***Модуль:*** *Tagged JSON*
*~~~~~~~~~~~*

*A compact representation for lossless serialization of non-standard JSON types. :class:`~flask.sessions.SecureCookieSessionInterface` uses this to serialize the session data, but it may be useful in other places. It can be extended to support other types.*

*.. autoclass:: TaggedJSONSerializer*
*:members:*

*.. autoclass:: JSONTag*
*:members:*

*Let's see an example that adds support for :class:`~collections.OrderedDict`. Dicts don't have an order in JSON, so to handle this we will dump the items as a list of ``[key, value]`` pairs. Subclass :class:`JSONTag` and give it the new key ``' od'`` to identify the type. The session serializer processes dicts first, so insert the new tag at the front of the order since ``OrderedDict`` must be processed before ``dict``.*

*.. code-block:: python*

```
from flask.json.tag import JSONTag

class TagOrderedDict(JSONTag):
    __slots__ = ('serializer',)
    key = ' od'

    def check(self, value):
        return isinstance(value, OrderedDict)

    def to_json(self, value):
        return [[k, self.serializer.tag(v)] for k, v in iteritems(value)]

    def to_python(self, value):
        return OrderedDict(value)
```

*app.session_interface.serializer.register(TagOrderedDict, index=0)*

**Классы:**

- **JSONTag**
  *Base class for defining type tags for :class:`TaggedJSONSerializer`.*
- **TagDict**
  *Tag for 1-item dicts whose only key matches a registered tag.*

  *Internally, the dict key is suffixed with `__`, and the suffix is removed when deserializing.*
- **PassDict**
- **TagTuple**
- **PassList**
- **TagBytes**
- **TagMarkup**
  *Serialize anything matching the :class:`~markupsafe.Markup` API by having a ``__html__`` method to the result of that method. Always deserializes to an instance of :class:`~markupsafe.Markup`.*
- **TagUUID**
- **TagDateTime**
- **TaggedJSONSerializer**
  *Serializer that uses a tag system to compactly represent objects that are not JSON types. Passed as the intermediate serializer to :class:`itsdangerous.Serializer`.*

  *The following extra types are supported:*

  ** :class:`dict`*
  ** :class:`tuple`*
  ** :class:`bytes`*
  ** :class:`~markupsafe.Markup`*
  ** :class:`~uuid.UUID`*
  ** :class:`~datetime.datetime`*

## src\flask\json__init__.py

**Функции:**

- **dumps**
  *Serialize data as JSON.*

  *If :data:`~flask.current_app` is available, it will use its*

*:meth:`app.json.dumps() <flask.json.provider.JSONProvider.dumps>`*
*method, otherwise it will use :func:`json.dumps`.*

*:param obj: The data to serialize.*
*:param kwargs: Arguments passed to the ``dumps`` implementation.*

*.. versionchanged:: 2.3*
*The ``app`` parameter was removed.*

*.. versionchanged:: 2.2*
*Calls ``current_app.json.dumps``, allowing an app to override*
*the behavior.*

*.. versionchanged:: 2.0.2*
*:class:`decimal.Decimal` is supported by converting to a string.*

*.. versionchanged:: 2.0*
*``encoding`` will be removed in Flask 2.1.*

*.. versionchanged:: 1.0.3*
*``app`` can be passed directly, rather than requiring an app*
*context for configuration.*

- **dump**
*Serialize data as JSON and write to a file.*

*If :data:`~flask.current_app` is available, it will use its*
*:meth:`app.json.dump() <flask.json.provider.JSONProvider.dump>`*
*method, otherwise it will use :func:`json.dump`.*

*:param obj: The data to serialize.*
*:param fp: A file opened for writing text. Should use the UTF-8*
*encoding to be valid JSON.*
*:param kwargs: Arguments passed to the ``dump`` implementation.*

*.. versionchanged:: 2.3*
*The ``app`` parameter was removed.*

*.. versionchanged:: 2.2*
*Calls ``current_app.json.dump``, allowing an app to override*
*the behavior.*

*.. versionchanged:: 2.0*
*Writing to a binary file, and the ``encoding`` argument, will be*
*removed in Flask 2.1.*

- **loads**
*Deserialize data as JSON.*

*If :data:`~flask.current_app` is available, it will use its*

*:meth:`app.json.loads() <flask.json.provider.JSONProvider.loads>` method, otherwise it will use :func:`json.loads`.*

*:param s: Text or UTF-8 bytes.*
*:param kwargs: Arguments passed to the ``loads`` implementation.*

*.. versionchanged:: 2.3*
*The ``app`` parameter was removed.*

*.. versionchanged:: 2.2*
*Calls ``current_app.json.loads``, allowing an app to override the behavior.*

*.. versionchanged:: 2.0*
*``encoding`` will be removed in Flask 2.1. The data must be a string or UTF-8 bytes.*

*.. versionchanged:: 1.0.3*
*``app`` can be passed directly, rather than requiring an app context for configuration.*

- **load**
*Deserialize data as JSON read from a file.*

*If :data:`~flask.current_app` is available, it will use its :meth:`app.json.load() <flask.json.provider.JSONProvider.load>` method, otherwise it will use :func:`json.load`.*

*:param fp: A file opened for reading text or UTF-8 bytes.*
*:param kwargs: Arguments passed to the ``load`` implementation.*

*.. versionchanged:: 2.3*
*The ``app`` parameter was removed.*

*.. versionchanged:: 2.2*
*Calls ``current_app.json.load``, allowing an app to override the behavior.*

*.. versionchanged:: 2.2*
*The ``app`` parameter will be removed in Flask 2.3.*

*.. versionchanged:: 2.0*
*``encoding`` will be removed in Flask 2.1. The file must be text mode, or binary mode with UTF-8 bytes.*

- **jsonify**
*Serialize the given arguments as JSON, and return a :class:`~flask.Response` object with the ``application/json`` mimetype. A dict or list returned from a view will be converted to a JSON response automatically without needing to call this.*

*This requires an active request or application context, and calls :meth:`app.json.response() <flask.json.provider.JSONProvider.response>`.*

*In debug mode, the output is formatted with indentation to make it easier to read. This may also be controlled by the provider.*

*Either positional or keyword arguments can be given, not both. If no arguments are given, ``None`` is serialized.*

*:param args: A single value to serialize, or multiple values to treat as a list to serialize.*
*:param kwargs: Treat as a dict to serialize.*

*.. versionchanged:: 2.2*
*Calls ``current_app.json.response``, allowing an app to override the behavior.*

*.. versionchanged:: 2.0.2*
*:class:`decimal.Decimal` is supported by converting to a string.*

*.. versionchanged:: 0.11*
*Added support for serializing top-level arrays. This was a security risk in ancient browsers. See :ref:`security-json`.*

*.. versionadded:: 0.2*

## src\flask\sansio\app.py

**Классы:**

- **App**
  *The flask object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.*

  *The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an :file:`__init__.py` file inside) or a standard module (just a ``.py`` file).*

  *For more information about resource loading, see :func:`open_resource`.*

  *Usually you create a :class:`Flask` instance in your main module or*

*in the :file:`__init__.py` file of your package like this::*

*from flask import Flask*
*app = Flask(__name__)*

*.. admonition:: About the First Parameter*

*The idea of the first parameter is to give Flask an idea of what belongs to your application. This name is used to find resources on the filesystem, can be used by extensions to improve debugging information and a lot more.*

*So it's important what you provide there. If you are using a single module, `__name__` is always the correct value. If you however are using a package, it's usually recommended to hardcode the name of your package there.*

*For example if your application is defined in :file:`yourapplication/app.py` you should create it with one of the two versions below::*

*app = Flask('yourapplication')*
*app = Flask(__name__.split('.')[0])*

*Why is that? The application will work even with `__name__`, thanks to how resources are looked up. However it will make debugging more painful. Certain extensions can make assumptions based on the import name of your application. For example the Flask-SQLAlchemy extension will look for the code in your application that triggered an SQL query in debug mode. If the import name is not properly set up, that debugging information is lost. (For example it would only pick up SQL queries in `yourapplication.app` and not `yourapplication.views.frontend`)*

*.. versionadded:: 0.7*
*The `static_url_path`, `static_folder`, and `template_folder` parameters were added.*

*.. versionadded:: 0.8*
*The `instance_path` and `instance_relative_config` parameters were added.*

*.. versionadded:: 0.11*
*The `root_path` parameter was added.*

*.. versionadded:: 1.0*
*The ``host_matching`` and ``static_host`` parameters were added.*

*.. versionadded:: 1.0*
*The ``subdomain_matching`` parameter was added. Subdomain*

*matching needs to be enabled manually now. Setting*
*:data:`SERVER_NAME` does not implicitly enable it.*

*:param import_name: the name of the application package*
*:param static_url_path: can be used to specify a different path for the*
*static files on the web. Defaults to the name*
*of the `static_folder` folder.*
*:param static_folder: The folder with static files that is served at*
*``static_url_path``. Relative to the application ``root_path``*
*or an absolute path. Defaults to ``'static'``.*
*:param static_host: the host to use when adding the static route.*
*Defaults to None. Required when using ``host_matching=True``*
*with a ``static_folder`` configured.*
*:param host_matching: set ``url_map.host_matching`` attribute.*
*Defaults to False.*
*:param subdomain_matching: consider the subdomain relative to*
*:data:`SERVER_NAME` when matching routes. Defaults to False.*
*:param template_folder: the folder that contains the templates that should*
*be used by the application. Defaults to*
*``'templates'`` folder in the root path of the*
*application.*
*:param instance_path: An alternative instance path for the application.*
*By default the folder ``'instance'`` next to the*
*package or module is assumed to be the instance*
*path.*
*:param instance_relative_config: if set to ``True`` relative filenames*
*for loading the config are assumed to*
*be relative to the instance path instead*
*of the application root.*
*:param root_path: The path to the root of the application files.*
*This should only be set manually when it can't be detected*
*automatically, such as for namespace packages.*

**Функции:**

- **_make_timedelta**

## src\flask\sansio\blueprints.py

**Классы:**

- **BlueprintSetupState**
  *Temporary holder object for registering a blueprint with the*
  *application. An instance of this class is created by the*

*:meth:`~flask.Blueprint.make_setup_state` method and later passed to all register callback functions.*

- **Blueprint**
  *Represents a blueprint, a collection of routes and other app-related functions that can be registered on a real application later.*

  *A blueprint is an object that allows defining application functions without requiring an application object ahead of time. It uses the same decorators as :class:`~flask.Flask`, but defers the need for an application by recording them for later registration.*

  *Decorating a function with a blueprint creates a deferred function that is called with :class:`~flask.blueprints.BlueprintSetupState` when the blueprint is registered on an application.*

  *See :doc:`/blueprints` for more information.*

  *:param name: The name of the blueprint. Will be prepended to each endpoint name.*
  *:param import_name: The name of the blueprint package, usually ``__name__``. This helps locate the ``root_path`` for the blueprint.*
  *:param static_folder: A folder with static files that should be served by the blueprint's static route. The path is relative to the blueprint's root path. Blueprint static files are disabled by default.*
  *:param static_url_path: The url to serve static files from. Defaults to ``static_folder``. If the blueprint does not have a ``url_prefix``, the app's static route will take precedence, and the blueprint's static files won't be accessible.*
  *:param template_folder: A folder with templates that should be added to the app's template search path. The path is relative to the blueprint's root path. Blueprint templates are disabled by default. Blueprint templates have a lower precedence than those in the app's templates folder.*
  *:param url_prefix: A path to prepend to all of the blueprint's URLs, to make them distinct from the rest of the app's routes.*
  *:param subdomain: A subdomain that blueprint routes will match on by default.*
  *:param url_defaults: A dict of default values that blueprint routes will receive by default.*
  *:param root_path: By default, the blueprint will automatically set this based on ``import_name``. In certain situations this automatic detection can fail, so the path can be specified manually instead.*

  *.. versionchanged:: 1.1.0*

*Blueprints have a ``cli`` group to register nested CLI commands.*
*The ``cli_group`` parameter controls the name of the group under*
*the ``flask`` command.*

*.. versionadded:: 0.7*

## src\flask\sansio\scaffold.py

**Классы:**

- **Scaffold**
  *Common behavior shared between :class:`~flask.Flask` and*
  *:class:`~flask.blueprints.Blueprint`.*

  *:param import_name: The import name of the module where this object*
  *is defined. Usually :attr:`__name__` should be used.*
  *:param static_folder: Path to a folder of static files to serve.*
  *If this is set, a static route will be added.*
  *:param static_url_path: URL prefix for the static route.*
  *:param template_folder: Path to a folder containing template files.*
  *for rendering. If this is set, a Jinja loader will be added.*
  *:param root_path: The path that static, template, and resource files*
  *are relative to. Typically not set, it is discovered based on*
  *the ``import_name``.*

  *.. versionadded:: 2.0*

**Функции:**

- **setupmethod**
- **_endpoint_from_view_func**
  *Internal helper that returns the default endpoint for a given*
  *function.  This always is the function name.*
- **_find_package_path**
  *Find the path that contains the package or module.*
- **find_package**
  *Find the prefix that a package is installed under, and the path*
  *that it would be imported from.*

  *The prefix is the directory containing the standard directory*
  *hierarchy (lib, bin, etc.). If the package is not installed to the*
  *system (:attr:`sys.prefix`) or a virtualenv (``site-packages``),*
  *``None`` is returned.*

*The path is the entry in :attr:`sys.path` that contains the package for import. If the package is not installed, it's assumed that the package was imported from the current working directory.*

## tests\conftest.py

**Функции:**

- **_standard_os_environ**
  *Set up ``os.environ`` at the start of the test session to have standard values. Returns a list of operations that is used by :func:`._reset_os_environ` after each test.*
- **_reset_os_environ**
  *Reset ``os.environ`` to the standard environ after each test, in case a test changed something without cleaning up.*
- **app**
- **app_ctx**
- **req_ctx**
- **client**
- **test_apps**
- **leak_detector**
- **modules_tmp_path**
  *A temporary directory added to sys.path.*
- **modules_tmp_path_prefix**
- **site_packages**
  *Create a fake site-packages.*
- **purge_module**

## tests\test_appctx.py

**Функции:**

- **test_basic_url_generation**
- **test_url_generation_requires_server_name**
- **test_url_generation_without_context_fails**
- **test_request_context_means_app_context**
- **test_app_context_provides_current_app**
- **test_app_tearing_down**
- **test_app_tearing_down_with_previous_exception**

- test_app_tearing_down_with_handled_exception_by_except_block
- test_app_tearing_down_with_handled_exception_by_app_handler
- test_app_tearing_down_with_unhandled_exception
- test_app_ctx_globals_methods
- test_custom_app_ctx_globals_class
- test_context_refcounts
- test_clean_pop

## tests\test_async.py

**Классы:**

- AppError
- BlueprintError
- AsyncView
- AsyncMethodView

**Функции:**

- _async_app
- test_async_route
- test_async_error_handler
- test_async_before_after_request

## tests\test_basic.py

**Функции:**

- test_options_work
- test_options_on_multiple_rules
- test_method_route
- test_method_route_no_methods
- test_provide_automatic_options_attr
- test_provide_automatic_options_kwarg
- test_request_dispatching
- test_disallow_string_for_allowed_methods
- test_url_mapping
- test_werkzeug_routing
- test_endpoint_decorator
- test_session
- test_session_path

- **test_session_using_application_root**
- **test_session_using_session_settings**
- **test_session_using_samesite_attribute**
- **test_missing_session**
- **test_session_secret_key_fallbacks**
- **test_session_expiration**
- **test_session_stored_last**
- **test_session_special_types**
- **test_session_cookie_setting**
- **test_session_vary_cookie**
- **test_session_refresh_vary**
- **test_flashes**
- **test_extended_flashing**
- **test_request_processing**
- **test_request_preprocessing_early_return**
- **test_after_request_processing**
- **test_teardown_request_handler**
- **test_teardown_request_handler_debug_mode**
- **test_teardown_request_handler_error**
- **test_before_after_request_order**
- **test_error_handling**
- **test_error_handling_processing**
- **test_baseexception_error_handling**
- **test_before_request_and_routing_errors**
- **test_user_error_handling**
- **test_http_error_subclass_handling**
- **test_errorhandler_precedence**
- **test_trap_bad_request_key_error**
- **test_trapping_of_all_http_exceptions**
- **test_error_handler_after_processor_error**
- **test_enctype_debug_helper**
- **test_response_types**
- **test_response_type_errors**
- **test_make_response**
- **test_make_response_with_response_instance**
- **test_jsonify_no_prettyprint**
- **test_jsonify_mimetype**
- **test_json_dump_dataclass**
- **test_jsonify_args_and_kwargs_check**
- **test_url_generation**
- **test_build_error_handler**
- **test_build_error_handler_reraise**
- **test_url_for_passes_special_values_to_build_error_handler**
- **test_static_files**
- **test_static_url_path**
- **test_static_url_path_with_ending_slash**
- **test_static_url_empty_path**
- **test_static_url_empty_path_default**
- **test_static_folder_with_pathlib_path**

- test_static_folder_with_ending_slash
- test_static_route_with_host_matching
- test_request_locals
- test_server_name_matching
- test_server_name_subdomain
- test_exception_propagation
- test_werkzeug_passthrough_errors
- test_url_processors
- test_inject_blueprint_url_defaults
- test_nonascii_pathinfo
- test_no_setup_after_first_request
- test_routing_redirect_debugging
- test_route_decorator_custom_endpoint
- test_get_method_on_g
- test_g_iteration_protocol
- test_subdomain_basic_support
- test_subdomain_matching
- test_subdomain_matching_with_ports
- test_subdomain_matching_other_name
- test_multi_route_rules
- test_multi_route_class_views
- test_run_defaults
- test_run_server_port
- test_run_from_config
- test_max_cookie_size
- test_app_freed_on_zero_refcount

## tests\test_blueprints.py

**Функции:**

- test_blueprint_specific_error_handling
- test_blueprint_specific_user_error_handling
- test_blueprint_app_error_handling
- test_blueprint_prefix_slash
- test_blueprint_url_defaults
- test_blueprint_url_processors
- test_templates_and_static
- test_default_static_max_age
- test_templates_list
- test_dotted_name_not_allowed
- test_empty_name_not_allowed
- test_dotted_names_from_app
- test_empty_url_defaults
- test_route_decorator_custom_endpoint

- **test_route_decorator_custom_endpoint_with_dots**
- **test_endpoint_decorator**
- **test_template_filter**
- **test_add_template_filter**
- **test_template_filter_with_name**
- **test_add_template_filter_with_name**
- **test_template_filter_with_template**
- **test_template_filter_after_route_with_template**
- **test_add_template_filter_with_template**
- **test_template_filter_with_name_and_template**
- **test_add_template_filter_with_name_and_template**
- **test_template_test**
- **test_add_template_test**
- **test_template_test_with_name**
- **test_add_template_test_with_name**
- **test_template_test_with_template**
- **test_template_test_after_route_with_template**
- **test_add_template_test_with_template**
- **test_template_test_with_name_and_template**
- **test_add_template_test_with_name_and_template**
- **test_context_processing**
- **test_template_global**
- **test_request_processing**
- **test_app_request_processing**
- **test_app_url_processors**
- **test_nested_blueprint**
- **test_nested_callback_order**
- **test_nesting_url_prefixes**
- **test_nesting_subdomains**
- **test_child_and_parent_subdomain**
- **test_unique_blueprint_names**
- **test_self_registration**
- **test_blueprint_renaming**

## tests\test_cli.py

**Классы:**

- **TestRoutes**

**Функции:**

- **runner**
- **test_cli_name**
  *Make sure the CLI object's name is the app's name and not the app itself*

- **test_find_best_app**
- **test_prepare_import**
  *Expect the correct path to be set and the correct import and app names to be returned.*

  *:func:`prepare_exec_for_file` has a side effect where the parent directory of the given import is added to :data:`sys.path`. This is reset after the test runs.*
- **test_locate_app**
- **test_locate_app_raises**
- **test_locate_app_suppress_raise**
- **test_get_version**
- **test_scriptinfo**
- **test_app_cli_has_app_context**
- **test_with_appcontext**
- **test_appgroup_app_context**
- **test_flaskgroup_app_context**
- **test_flaskgroup_debug**
- **test_flaskgroup_nested**
- **test_no_command_echo_loading_error**
- **test_help_echo_loading_error**
- **test_help_echo_exception**
- **dotenv_not_available**
- **test_load_dotenv**
- **test_dotenv_path**
- **test_dotenv_optional**
- **test_disable_dotenv_from_env**
- **test_run_cert_path**
- **test_run_cert_adhoc**
- **test_run_cert_import**
- **test_run_cert_no_ssl**
- **test_cli_blueprints**
  *Test blueprint commands register correctly to the application*
- **test_cli_empty**
  *If a Blueprint's CLI group is empty, do not register it.*
- **test_run_exclude_patterns**

## tests\test_config.py

**Функции:**

- **common_object_test**
- **test_config_from_pyfile**

- **test_config_from_object**
- **test_config_from_file_json**
- **test_config_from_file_toml**
- **test_from_prefixed_env**
- **test_from_prefixed_env_custom_prefix**
- **test_from_prefixed_env_nested**
- **test_config_from_mapping**
- **test_config_from_class**
- **test_config_from_envvar**
- **test_config_from_envvar_missing**
- **test_config_missing**
- **test_config_missing_file**
- **test_custom_config_class**
- **test_session_lifetime**
- **test_get_namespace**
- **test_from_pyfile_weird_encoding**

## tests\test_converters.py

**Функции:**

- **test_custom_converters**
- **test_context_available**

## tests\test_helpers.py

**Классы:**

- **FakePath**
  *Fake object to represent a ``PathLike object``.*

  *This represents a ``pathlib.Path`` object in python 3.*
  *See: https://www.python.org/dev/peps/pep-0519/*
- **PyBytesIO**
- **TestSendfile**
- **TestUrlFor**
- **TestNoImports**
  *Test Flasks are created without import.*

  *Avoiding ``__import__`` helps create Flask instances where there are errors*

*at import time.  Those runtime errors will be apparent to the user soon enough, but tools which build Flask instances meta-programmatically benefit*
*from a Flask which does not ``__import__``.  Instead of importing to retrieve file paths or metadata on a module or package, use the pkgutil and imp modules in the Python standard library.*

- **TestStreaming**
- **TestHelpers**

**Функции:**

- **test_redirect_no_app**
- **test_redirect_with_app**
- **test_abort_no_app**
- **test_app_aborter_class**
- **test_abort_with_app**
- **test_open_resource**
- **test_open_resource_exceptions**
- **test_open_resource_with_encoding**

## tests\test_instance_config.py

**Функции:**

- **test_explicit_instance_paths**
- **test_uninstalled_module_paths**
- **test_uninstalled_package_paths**
- **test_uninstalled_namespace_paths**
- **test_installed_module_paths**
- **test_installed_package_paths**
- **test_prefix_package_paths**

## tests\test_json.py

**Классы:**

- **FixedOffset**
  *Fixed offset in hours east from UTC.*

  *This is a slight adaptation of the ``FixedOffset`` example found in https://docs.python.org/2.7/library/datetime.html.*

**Функции:**

- **test_bad_request_debug_message**
- **test_json_bad_requests**
- **test_json_custom_mimetypes**
- **test_json_as_unicode**
- **test_json_dump_to_file**
- **test_jsonify_basic_types**
- **test_jsonify_dicts**
- **test_jsonify_arrays**
  *Test jsonify of lists and args unpacking.*
- **test_jsonify_datetime**
- **test_jsonify_aware_datetimes**
  *Test if aware datetime.datetime objects are converted into GMT.*
- **test_jsonify_uuid_types**
  *Test jsonify with uuid.UUID types*
- **test_json_decimal**
- **test_json_attr**
- **test_tojson_filter**
- **test_json_customization**
- **_has_encoding**
- **test_json_key_sorting**
- **test_html_method**

## tests\test_json_tag.py

**Функции:**

- **test_dump_load_unchanged**
- **test_duplicate_tag**
- **test_custom_tag**
- **test_tag_interface**
- **test_tag_order**

## tests\test_logging.py

**Функции:**

- **reset_logging**
- **test_logger**

- test_logger_debug
- test_existing_handler
- test_wsgi_errors_stream
- test_has_level_handler
- test_log_view_exception

## tests\test_regression.py

**Функции:**

- test_aborting

## tests\test_reqctx.py

**Классы:**

- TestGreenletContextCopying

**Функции:**

- test_teardown_on_pop
- test_teardown_with_previous_exception
- test_teardown_with_handled_exception
- test_proper_test_request_context
- test_context_binding
- test_context_test
- test_manual_context_binding
- test_session_error_pops_context
- test_session_dynamic_cookie_name
- test_bad_environ_raises_bad_request
- test_environ_for_valid_idna_completes
- test_normal_environ_completes

## tests\test_request.py

**Функции:**

- **test_max_content_length**
- **test_limit_config**
- **test_trusted_hosts_config**

## tests\test_session_interface.py

**Функции:**

- **test_open_session_with_endpoint**
  *If request.endpoint (or other URL matching behavior) is needed while loading the session, RequestContext.match_request() can be called manually.*

## tests\test_signals.py

**Функции:**

- **test_template_rendered**
- **test_before_render_template**
- **test_request_signals**
- **test_request_exception_signal**
- **test_appcontext_signals**
- **test_flash_signal**
- **test_appcontext_tearing_down_signal**

## tests\test_subclassing.py

**Функции:**

- **test_suppressed_exception_logging**

## tests\test_templating.py

**Функции:**

- **test_context_processing**
- **test_original_win**
- **test_simple_stream**
- **test_request_less_rendering**
- **test_standard_context**
- **test_escaping**
- **test_no_escaping**
- **test_escaping_without_template_filename**
- **test_macros**
- **test_template_filter**
- **test_add_template_filter**
- **test_template_filter_with_name**
- **test_add_template_filter_with_name**
- **test_template_filter_with_template**
- **test_add_template_filter_with_template**
- **test_template_filter_with_name_and_template**
- **test_add_template_filter_with_name_and_template**
- **test_template_test**
- **test_add_template_test**
- **test_template_test_with_name**
- **test_add_template_test_with_name**
- **test_template_test_with_template**
- **test_add_template_test_with_template**
- **test_template_test_with_name_and_template**
- **test_add_template_test_with_name_and_template**
- **test_add_template_global**
- **test_custom_template_loader**
- **test_iterable_loader**
- **test_templates_auto_reload**
- **test_templates_auto_reload_debug_run**
- **test_template_loader_debugging**

- **test_custom_jinja_env**

## tests\test_testing.py

**Функции:**

- **test_environ_defaults_from_config**
- **test_environ_defaults**
- **test_environ_base_default**
- **test_environ_base_modified**
- **test_client_open_environ**
- **test_specify_url_scheme**
- **test_path_is_url**
- **test_environbuilder_json_dumps**
  *EnvironBuilder.json_dumps() takes settings from the app.*
- **test_blueprint_with_subdomain**
- **test_redirect_keep_session**
- **test_session_transactions**
- **test_session_transactions_no_null_sessions**
- **test_session_transactions_keep_context**
- **test_session_transaction_needs_cookies**
- **test_test_client_context_binding**
- **test_reuse_client**
- **test_full_url_request**
- **test_json_request_and_response**
- **test_client_json_no_app_context**
- **test_subdomain**
- **test_nosubdomain**
- **test_cli_runner_class**
- **test_cli_invoke**
- **test_cli_custom_obj**
- **test_client_pop_all_preserved**

## tests\test_user_error_handler.py

**Классы:**

- **TestGenericHandlers**
  *Test how very generic handlers are dispatched to.*

**Функции:**

- test_error_handler_no_match
- test_error_handler_subclass
- test_error_handler_http_subclass
- test_error_handler_blueprint
- test_default_error_handler

## tests\test_views.py

**Функции:**

- common_test
- test_basic_view
- test_method_based_view
- test_view_patching
- test_view_inheritance
- test_view_decorators
- test_view_provide_automatic_options_attr
- test_implicit_head
- test_explicit_head
- test_endpoint_override
- test_methods_var_inheritance
- test_multiple_inheritance
- test_remove_method_from_parent
- test_init_once

## tests\test_apps\blueprintapp__init__.py

## tests\test_apps\blueprintapp\apps__init__.py

## tests\test_apps\blueprintapp\apps\admin__init__.py

**Функции:**

- **index**
- **index2**

**API Endpoints:**

| Function | Decorator | Path |
|---|---|---|
| index | route | / |
| index2 | route | /index2 |

## tests\test_apps\blueprintapp\apps\frontend__init__.py

**Функции:**

- **index**
- **missing_template**

**API Endpoints:**

| Function | Decorator | Path |
|---|---|---|
| index | route | / |
| missing_template | route | /missing |

## tests\test_apps\cliapp\app.py

## tests\test_apps\cliapp\factory.py

**Функции:**

- **create_app**
- **create_app2**
- **no_app**

## tests\test_apps\cliapp\importerrorapp.py

## tests\test_apps\cliapp\multiapp.py

## tests\test_apps\cliapp__init__.py

## tests\test_apps\cliapp\inner1__init__.py

## tests\test_apps\cliapp\inner1\inner2\flask.py

## tests\test_apps\cliapp\inner1\inner2__init__.py

## tests\test_apps\helloworld\hello.py

**Функции:**

- **hello**

**API Endpoints:**

| Function | Decorator | Path |
| --- | --- | --- |

| | | |
|---|---|---|
| hello | route | / |

## tests\test_apps\helloworld\wsgi.py

## tests\test_apps\subdomaintestmodule__init__.py

## tests\type_check\typing_app_decorators.py

**Функции:**

- **after_sync**
- **before_sync**
- **teardown_sync**

## tests\type_check\typing_error_handler.py

**Функции:**

- **handle_400**
- **handle_custom**
- **handle_accept_base**
- **handle_multiple**

## tests\type_check\typing_route.py

**Классы:**

- **StatusJSON**

- **RenderTemplateView**

**Функции:**

- **hello_str**
- **hello_bytes**
- **hello_json**
- **hello_json_dict**
- **hello_json_list**
- **typed_dict**
- **hello_generator**
- **hello_generator_expression**
- **hello_iterator**
- **tuple_status**
- **tuple_status_enum**
- **tuple_headers**
- **return_template**
- **return_template_stream**

**API Endpoints:**

| Function | Decorator | Path |
|---|---|---|
| hello_str | route | /str |
| hello_bytes | route | /bytes |
| hello_json | route | /json |
| hello_json_dict | route | /json/dict |
| hello_json_list | route | /json/dict |
| typed_dict | route | /typed-dict |
| hello_generator | route | /generator |
| hello_generator_expression | route | /generator-expression |
| hello_iterator | route | /iterator |
| tuple_status | route | /status |
| tuple_status | route | /status/<int:code> |
| tuple_status_enum | route | /status-enum |
| tuple_headers | route | /headers |
| return_template | route | /template |
| return_template | route | /template/<name> |
| return_template_stream | route | /template |

## Последние коммиты

**2025-03-30 13:17** - *David Lord*
Merge branch 'stable'

**2025-03-30 13:15** - *David Lord*
Async Iterable Response (#5659)

**2025-03-30 13:14** - *CoolCat467*
Accept `AsyncIterable` for responses

**2025-03-29 15:57** - *David Lord*
add ghsa links

**2025-03-29 15:45** - *David Lord*
remove tests about deprecated pkgutil.get_loader (#5702)

**2025-03-29 15:42** - *David Lord*
remove tests about deprecated pkgutil.get_loader

**2025-03-29 15:32** - *David Lord*
add endpoint name in favicon example (#5701)

**2025-03-29 15:30** - *David Lord*
add endpoint name in favicon example

**2025-03-29 15:28** - *David Lord*
better type checking during deprecation (#5700)

**2025-03-29 15:23** - *David Lord*
better type checking during deprecation