

Changelog

Version 1.1.2

Unreleased

- Work around an issue when running the `flask` command with an external debugger on Windows. [#3297](#)
- The static route will not catch all URLs if the `Flask static_folder` argument ends with a slash. [#3452](#)

Version 1.1.1

Released 2019-07-08

- The `flask.json_available` flag was added back for compatibility with some extensions. It will raise a deprecation warning when used, and will be removed in version 2.0.0. [#3288](#)

Version 1.1.0

Released 2019-07-04

- Bump minimum Werkzeug version to ≥ 0.15 .
- Drop support for Python 3.4.
- Error handlers for `InternalServerError` or `500` will always be passed an instance of `InternalServerError`. If they are invoked due to an unhandled exception, that original exception is now available as `e.original_exception` rather than being passed directly to the handler. The same is true if the handler is for the base `HTTPException`. This makes error handler behavior more consistent. [#3266](#)
 - `Flask.finalize_request()` is called for all unhandled exceptions even if there is no `500` error handler.
- `Flask.logger` takes the same name as `Flask.name` (the value passed as `Flask(import_name)`). This reverts 1.0's behavior of always logging to `"flask.app"`, in order to support multiple apps in the same process. A warning will be shown if old configuration is detected that needs to be moved. [#2866](#)
- `flask.RequestContext.copy()` includes the current session object in the context copy. This prevents `session` pointing to an out-of-date object. [#2935](#)

- Using built-in RequestContext, unprintable Unicode characters in Host header will result in a HTTP 400 response and not HTTP 500 as previously. [#2994](#)
- `send_file()` supports `PathLike` objects as described in PEP 0519, to support `pathlib` in Python 3. [#3059](#)
- `send_file()` supports `BytesIO` partial content. [#2957](#)
- `open_resource()` accepts the “rt” file mode. This still does the same thing as “r”. [#3163](#)
- The `MethodView.methods` attribute set in a base class is used by subclasses. [#3138](#)
- `Flask.jinja_options` is a `dict` instead of an `ImmutableDict` to allow easier configuration. Changes must still be made before creating the environment. [#3190](#)
- Flask’s `JSONMixin` for the request and response wrappers was moved into Werkzeug. Use Werkzeug’s version with Flask-specific support. This bumps the Werkzeug dependency to ≥ 0.15 . [#3125](#)
- The `flask` command entry point is simplified to take advantage of Werkzeug 0.15’s better reloader support. This bumps the Werkzeug dependency to ≥ 0.15 . [#3022](#)
- Support `static_url_path` that ends with a forward slash. [#3134](#)
- Support empty `static_folder` without requiring setting an empty `static_url_path` as well. [#3124](#)
- `jsonify()` supports `dataclasses.dataclass` objects. [#3195](#)
- Allow customizing the `Flask.url_map_class` used for routing. [#3069](#)
- The development server port can be set to 0, which tells the OS to pick an available port. [#2926](#)
- The return value from `cli.load_dotenv()` is more consistent with the documentation. It will return `False` if python-dotenv is not installed, or if the given path isn’t a file. [#2937](#)
- Signaling support has a stub for the `connect_via` method when the Blinker library is not installed. [#3208](#)
- Add an `--extra-files` option to the `flask run` CLI command to specify extra files that will trigger the reloader on change. [#2897](#)
- Allow returning a dictionary from a view function. Similar to how returning a string will produce a `text/html` response, returning a dict will call `jsonify` to produce a `application/json` response. [#3111](#)
- Blueprints have a `cli` Click group like `app.cli`. CLI commands registered with a blueprint will be available as a group under the `flask` command. [#1357](#)
- When using the test client as a context manager (`with client:`), all preserved request contexts are popped when the block exits, ensuring nested contexts are cleaned up correctly. [#3157](#)
- Show a better error message when the view return type is not supported. [#3214](#)
- `flask.testing.make_test_environ_builder()` has been deprecated in favour of a new class `flask.testing.EnvironBuilder`. [#3232](#)
- The `flask run` command no longer fails if Python is not built with SSL support. Using the `--cert` option will show an appropriate error message. [#3211](#)

- URL matching now occurs after the request context is pushed, rather than when it's created. This allows custom URL converters to access the app and request contexts, such as to query a database for an id. [#3088](#)

Version 1.0.4

Released 2019-07-04

- The key information for `BadRequestKeyError` is no longer cleared outside debug mode, so error handlers can still access it. This requires upgrading to Werkzeug 0.15.5. [#3249](#)
- `send_file` url quotes the “.” and “/” characters for more compatible UTF-8 filename support in some browsers. [#3074](#)
- Fixes for PEP451 import loaders and pytest 5.x. [#3275](#)
- Show message about dotenv on stderr instead of stdout. [#3285](#)

Version 1.0.3

Released 2019-05-17

- `send_file()` encodes filenames as ASCII instead of Latin-1 (ISO-8859-1). This fixes compatibility with Gunicorn, which is stricter about header encodings than PEP 3333. [#2766](#)
- Allow custom CLIs using `FlaskGroup` to set the debug flag without it always being overwritten based on environment variables. [#2765](#)
- `flask --version` outputs Werkzeug's version and simplifies the Python version. [#2825](#)
- `send_file()` handles an `attachment_filename` that is a native Python 2 string (bytes) with UTF-8 coded bytes. [#2933](#)
- A catch-all error handler registered for `HTTPException` will not handle `RoutingException`, which is used internally during routing. This fixes the unexpected behavior that had been introduced in 1.0. [#2986](#)
- Passing the `json` argument to `app.test_client` does not push/pop an extra app context. [#2900](#)

Version 1.0.2

Released 2018-05-02

- Fix more backwards compatibility issues with merging slashes between a blueprint prefix and route. [#2748](#)

- Fix error with `flask routes` command when there are no routes. [#2751](#)

Version 1.0.1

Released 2018-04-29

- Fix registering partials (with no `__name__`) as view functions. [#2730](#)
- Don't treat lists returned from view functions the same as tuples. Only tuples are interpreted as response data. [#2736](#)
- Extra slashes between a blueprint's `url_prefix` and a route URL are merged. This fixes some backwards compatibility issues with the change in 1.0. [#2731](#), [#2742](#)
- Only trap `BadRequestKeyError` errors in debug mode, not all `BadRequest` errors. This allows `abort(400)` to continue working as expected. [#2735](#)
- The `FLASK_SKIP_DOTENV` environment variable can be set to `1` to skip automatically loading dotenv files. [#2722](#)

Version 1.0

Released 2018-04-26

- Python 2.6 and 3.3 are no longer supported.
- Bump minimum dependency versions to the latest stable versions: Werkzeug `>= 0.14`, Jinja `>= 2.10`, itsdangerous `>= 0.24`, Click `>= 5.1`. [#2586](#)
- Skip `app.run` when a Flask application is run from the command line. This avoids some behavior that was confusing to debug.
- Change the default for `JSONIFY_PRETTYPRINT_REGULAR` to `False`. `jsonify()` returns a compact format by default, and an indented format in debug mode. [#2193](#)
- `Flask.__init__` accepts the `host_matching` argument and sets it on `url_map`. [#1559](#)
- `Flask.__init__` accepts the `static_host` argument and passes it as the `host` argument when defining the static route. [#1559](#)
- `send_file()` supports Unicode in `attachment_filename`. [#2223](#)
- Pass `_scheme` argument from `url_for()` to `handle_url_build_error()`. [#2017](#)
- `add_url_rule()` accepts the `provide_automatic_options` argument to disable adding the `OPTIONS` method. [#1489](#)
- `MethodView` subclasses inherit method handlers from base classes. [#1936](#)
- Errors caused while opening the session at the beginning of the request are handled by the app's error handlers. [#2254](#)
- Blueprints gained `json_encoder` and `json_decoder` attributes to override the app's encoder and decoder. [#1898](#)
- `Flask.make_response()` raises `TypeError` instead of `ValueError` for bad response types. The error messages have been improved to describe why the type is invalid.

[#2256](#)

- Add `routes` CLI command to output routes registered on the application. [#2259](#)
- Show warning when session cookie domain is a bare hostname or an IP address, as these may not behave properly in some browsers, such as Chrome. [#2282](#)
- Allow IP address as exact session cookie domain. [#2282](#)
- `SESSION_COOKIE_DOMAIN` is set if it is detected through `SERVER_NAME`. [#2282](#)
- Auto-detect zero-argument app factory called `create_app` or `make_app` from `FLASK_APP`. [#2297](#)
- Factory functions are not required to take a `script_info` parameter to work with the `flask` command. If they take a single parameter or a parameter named `script_info`, the `ScriptInfo` object will be passed. [#2319](#)
- `FLASK_APP` can be set to an app factory, with arguments if needed, for example `FLASK_APP=myproject.app:create_app('dev')`. [#2326](#)
- `FLASK_APP` can point to local packages that are not installed in editable mode, although `pip install -e` is still preferred. [#2414](#)
- The `View` class attribute `provide_automatic_options` is set in `as_view()`, to be detected by `add_url_rule()`. [#2316](#)
- Error handling will try handlers registered for `blueprint, code, app, code, blueprint, exception, app, exception`. [#2314](#)
- `Cookie` is added to the response's `Vary` header if the session is accessed at all during the request (and not deleted). [#2288](#)
- `test_request_context()` accepts `subdomain` and `url_scheme` arguments for use when building the base URL. [#1621](#)
- Set `APPLICATION_ROOT` to `'/'` by default. This was already the implicit default when it was set to `None`.
- `TRAP_BAD_REQUEST_ERRORS` is enabled by default in debug mode. `BadRequestKeyError` has a message with the bad key in debug mode instead of the generic bad request message. [#2348](#)
- Allow registering new tags with `TaggedJSONSerializer` to support storing other types in the session cookie. [#2352](#)
- Only open the session if the request has not been pushed onto the context stack yet. This allows `stream_with_context()` generators to access the same session that the containing view uses. [#2354](#)
- Add `json` keyword argument for the test client request methods. This will dump the given object as JSON and set the appropriate content type. [#2358](#)
- Extract JSON handling to a mixin applied to both the `Request` and `Response` classes. This adds the `is_json()` and `get_json()` methods to the response to make testing JSON response much easier. [#2358](#)
- Removed error handler caching because it caused unexpected results for some exception inheritance hierarchies. Register handlers explicitly for each exception to avoid traversing the MRO. [#2362](#)
- Fix incorrect JSON encoding of aware, non-UTC datetimes. [#2374](#)

- Template auto reloading will honor debug mode even even if `jinja_env` was already accessed. [#2373](#)
- The following old deprecated code was removed. [#2385](#)
 - `flask.ext` - import extensions directly by their name instead of through the `flask.ext` namespace. For example, `import flask.ext.sqlalchemy` becomes `import flask_sqlalchemy`.
 - `Flask.init_jinja_globals` - extend `Flask.create_jinja_environment()` instead.
 - `Flask.error_handlers` - tracked by `Flask.error_handler_spec`, use `Flask.errorhandler()` to register handlers.
 - `Flask.request_globals_class` - use `Flask.app_ctx_globals_class` instead.
 - `Flask.static_path` - use `Flask.static_url_path` instead.
 - `Request.module` - use `Request.blueprint` instead.
- The `Request.json` property is no longer deprecated. [#1421](#)
- Support passing a `EnvironBuilder` or `dict` to `test_client.open`. [#2412](#)
- The `flask` command and `Flask.run()` will load environment variables from `.env` and `.flaskenv` files if python-dotenv is installed. [#2416](#)
- When passing a full URL to the test client, the scheme in the URL is used instead of `PREFERRED_URL_SCHEME`. [#2430](#)
- `Flask.logger` has been simplified. `LOGGER_NAME` and `LOGGER_HANDLER_POLICY` config was removed. The logger is always named `flask.app`. The level is only set on first access, it doesn't check `Flask.debug` each time. Only one format is used, not different ones depending on `Flask.debug`. No handlers are removed, and a handler is only added if no handlers are already configured. [#2436](#)
- Blueprint view function names may not contain dots. [#2450](#)
- Fix a `ValueError` caused by invalid `Range` requests in some cases. [#2526](#)
- The development server uses threads by default. [#2529](#)
- Loading config files with `silent=True` will ignore `ENOTDIR` errors. [#2581](#)
- Pass `--cert` and `--key` options to `flask run` to run the development server over HTTPS. [#2606](#)
- Added `SESSION_COOKIE_SAMESITE` to control the `SameSite` attribute on the session cookie. [#2607](#)
- Added `test_cli_runner()` to create a Click runner that can invoke Flask CLI commands for testing. [#2636](#)
- Subdomain matching is disabled by default and setting `SERVER_NAME` does not implicitly enable it. It can be enabled by passing `subdomain_matching=True` to the `Flask` constructor. [#2635](#)
- A single trailing slash is stripped from the blueprint `url_prefix` when it is with the app. [#2629](#)

- `Request.get_json()` doesn't cache the result if parsing fails when `silent` is true. [#2651](#)
- `Request.get_json()` no longer accepts arbitrary encodings. Incoming JSON should be encoded using UTF-8 per [RFC 8259](#), but Flask will autodetect UTF-8, -16, or -32. [#2691](#)
- Added `MAX_COOKIE_SIZE` and `Response.max_cookie_size` to control when Werkzeug warns about large cookies that browsers may ignore. [#2693](#)
- Updated documentation theme to make docs look better in small windows. [#2709](#)
- Rewrote the tutorial docs and example project to take a more structured approach to help new users avoid common pitfalls. [#2676](#)

Version 0.12.5

Released 2020-02-10

- Pin Werkzeug to < 1.0.0. [#3497](#)

Version 0.12.4

Released 2018-04-29

- Repackage 0.12.3 to fix package layout issue. [#2728](#)

Version 0.12.3

Released 2018-04-26

- `Request.get_json()` no longer accepts arbitrary encodings. Incoming JSON should be encoded using UTF-8 per [RFC 8259](#), but Flask will autodetect UTF-8, -16, or -32. [#2692](#)
- Fix a Python warning about imports when using `python -m flask`. [#2666](#)
- Fix a `ValueError` caused by invalid `Range` requests in some cases.

Version 0.12.2

Released 2017-05-16

- Fix a bug in `safe_join` on Windows.

 v: 1.1.x ▼

Version 0.12.1

Released 2017-03-31

- Prevent `flask run` from showing a `NoAppException` when an `ImportError` occurs within the imported application module.
- Fix encoding behavior of `app.config.from_pyfile` for Python 3. [#2118](#)
- Use the `SERVER_NAME` config if it is present as default values for `app.run`. [#2109](#), [#2152](#)
- Call `ctx.auto_pop` with the exception object instead of `None`, in the event that a `BaseException` such as `KeyboardInterrupt` is raised in a request handler.

Version 0.12

Released 2016-12-21, codename Punsch

- The cli command now responds to `--version`.
- Mimetype guessing and ETag generation for file-like objects in `send_file` has been removed. [#104](#), [:pr`1849`](#)
- Mimetype guessing in `send_file` now fails loudly and doesn't fall back to `application/octet-stream`. [#1988](#)
- Make `flask.safe_join` able to join multiple paths like `os.path.join` [#1730](#)
- Revert a behavior change that made the dev server crash instead of returning an Internal Server Error. [#2006](#)
- Correctly invoke response handlers for both regular request dispatching as well as error handlers.
- Disable logger propagation by default for the app logger.
- Add support for range requests in `send_file`.
- `app.test_client` includes preset default environment, which can now be directly set, instead of per `client.get`.
- Fix crash when running under PyPy3. [#1814](#)

Version 0.11.1

Released 2016-06-07

- Fixed a bug that prevented `FLASK_APP=foobar/__init__.py` from working. [#1872](#)

Version 0.11

Released 2016-05-29, codename Absinthe

 v: 1.1.x ▼

- Added support to serializing top-level arrays to `flask jsonify()`. This introduces a security risk in ancient browsers. See [JSON Security](#) for details.


- Added `before_render_template` signal.
- Added `**kwargs` to `flask.Test.test_client()` to support passing additional keyword arguments to the constructor of `flask.Flask.test_client_class`.
- Added `SESSION_REFRESH_EACH_REQUEST` config key that controls the set-cookie behavior. If set to `True` a permanent session will be refreshed each request and get their lifetime extended, if set to `False` it will only be modified if the session actually modifies. Non permanent sessions are not affected by this and will always expire if the browser window closes.
- Made Flask support custom JSON mimetypes for incoming data.
- Added support for returning tuples in the form `(response, headers)` from a view function.
- Added `flask.Config.from_json()`.
- Added `flask.Flask.config_class`.
- Added `flask.Config.get_namespace()`.
- Templates are no longer automatically reloaded outside of debug mode. This can be configured with the new `TEMPLATES_AUTO_RELOAD` config key.
- Added a workaround for a limitation in Python 3.3's namespace loader.
- Added support for explicit root paths when using Python 3.3's namespace packages.
- Added `flask` and the `flask.cli` module to start the local debug server through the click CLI system. This is recommended over the old `flask.run()` method as it works faster and more reliable due to a different design and also replaces `Flask-Script`.
- Error handlers that match specific classes are now checked first, thereby allowing catching exceptions that are subclasses of HTTP exceptions (in `werkzeug.exceptions`). This makes it possible for an extension author to create exceptions that will by default result in the HTTP error of their choosing, but may be caught with a custom error handler if desired.
- Added `flask.Config.from_mapping()`.
- Flask will now log by default even if debug is disabled. The log format is now hardcoded but the default log handling can be disabled through the `LOGGER_HANDLER_POLICY` configuration key.
- Removed deprecated module functionality.
- Added the `EXPLAIN_TEMPLATE_LOADING` config flag which when enabled will instruct Flask to explain how it locates templates. This should help users debug when the wrong templates are loaded.
- Enforce blueprint handling in the order they were registered for template loading.
- Ported test suite to `py.test`.
- Deprecated `request.json` in favour of `request.get_json()`.
- Add “pretty” and “compressed” separators definitions in `jsonify()` method. Reduces JSON response size when `JSONIFY_PRETTYPRINT_REGULAR=False` by removing unnecessary white space included by default after separators.
- JSON responses are now terminated with a newline character, because it is a convention that UNIX text files end with a newline and some clients don't deal well when this

newline is missing. This came up originally as a part of <https://github.com/postman-labs/httpbin/issues/168>. [#1262](#)

- The automatically provided `OPTIONS` method is now correctly disabled if the user registered an overriding rule with the lowercase-version `options`. [#1288](#)
- `flask.json jsonify` now supports the `datetime.date` type. [#1326](#)
- Don't leak exception info of already caught exceptions to context teardown handlers. [#1393](#)
- Allow custom Jinja environment subclasses. [#1422](#)
- Updated extension dev guidelines.
- `flask.g` now has `pop()` and `setdefault` methods.
- Turn on autoescape for `flask.templating.render_template_string` by default. [#1515](#)
- `flask.ext` is now deprecated. [#1484](#)
- `send_from_directory` now raises `BadRequest` if the filename is invalid on the server OS. [#1763](#)
- Added the `JSONIFY_MIMETYPE` configuration variable. [#1728](#)
- Exceptions during teardown handling will no longer leave bad application contexts lingering around.
- Fixed broken `test_appcontext_signals()` test case.
- Raise an `AttributeError` in `flask.helpers.find_package()` with a useful message explaining why it is raised when a PEP 302 import hook is used without an `is_package()` method.
- Fixed an issue causing exceptions raised before entering a request or app context to be passed to teardown handlers.
- Fixed an issue with query parameters getting removed from requests in the test client when absolute URLs were requested.
- Made `@before_first_request` into a decorator as intended.
- Fixed an etags bug when sending a file streams with a name.
- Fixed `send_from_directory` not expanding to the application root path correctly.
- Changed logic of before first request handlers to flip the flag after invoking. This will allow some uses that are potentially dangerous but should probably be permitted.
- Fixed Python 3 bug when a handler from `app.url_build_error_handlers` reraises the `BuildError`.

Version 0.10.1


Released 2013-06-14

- Fixed an issue where `|tojson` was not quoting single quotes which made the filter not work properly in HTML attributes. Now it's possible to use that filter in sin  `v: 1.1.x` ▼ attributes. This should make using that filter with angular.js easier.

- Added support for byte strings back to the session system. This broke compatibility with the common case of people putting binary data for token verification into the session.
- Fixed an issue where registering the same method twice for the same endpoint would trigger an exception incorrectly.

Version 0.10

Released 2013-06-13, codename Limoncello

- Changed default cookie serialization format from pickle to JSON to limit the impact an attacker can do if the secret key leaks. See [Version 0.10](#) for more information.
- Added `template_test` methods in addition to the already existing `template_filter` method family.
- Added `template_global` methods in addition to the already existing `template_filter` method family.
- Set the content-length header for x-sendfile.
- `tojson` filter now does not escape script blocks in HTML5 parsers.
- `tojson` used in templates is now safe by default due. This was allowed due to the different escaping behavior.
- Flask will now raise an error if you attempt to register a new function on an already used endpoint.
- Added wrapper module around simplejson and added default serialization of datetime objects. This allows much easier customization of how JSON is handled by Flask or any Flask extension.
- Removed deprecated internal `flask.session` module alias. Use `flask.sessions` instead to get the session module. This is not to be confused with `flask.session` the session proxy.
- Templates can now be rendered without request context. The behavior is slightly different as the `request`, `session` and `g` objects will not be available and blueprint's context processors are not called.
- The config object is now available to the template as a real global and not through a context processor which makes it available even in imported templates by default.
- Added an option to generate non-ascii encoded JSON which should result in less bytes being transmitted over the network. It's disabled by default to not cause confusion with existing libraries that might expect `flask.json.dumps` to return bytestrings by default.
- `flask.g` is now stored on the app context instead of the request context.
- `flask.g` now gained a `get()` method for not erroring out on non existing items.
- `flask.g` now can be used with the `in` operator to see what's defined and it  `v: 1.1.x` ▼
able and will yield all attributes stored.

- `flask.Flask.request_globals_class` got renamed to `flask.Flask.app_ctx_globals_class` which is a better name to what it does since 0.10.
- `request`, `session` and `g` are now also added as proxies to the template context which makes them available in imported templates. One has to be very careful with those though because usage outside of macros might cause caching.
- Flask will no longer invoke the wrong error handlers if a proxy exception is passed through.
- Added a workaround for chrome's cookies in localhost not working as intended with domain names.
- Changed logic for picking defaults for cookie values from sessions to work better with Google Chrome.
- Added `message_flashed` signal that simplifies flashing testing.
- Added support for copying of request contexts for better working with greenlets.
- Removed custom JSON HTTP exception subclasses. If you were relying on them you can reintroduce them again yourself trivially. Using them however is strongly discouraged as the interface was flawed.
- Python requirements changed: requiring Python 2.6 or 2.7 now to prepare for Python 3.3 port.
- Changed how the teardown system is informed about exceptions. This is now more reliable in case something handles an exception halfway through the error handling process.
- Request context preservation in debug mode now keeps the exception information around which means that teardown handlers are able to distinguish error from success cases.
- Added the `JSONIFY_PRETTYPRINT_REGULAR` configuration variable.
- Flask now orders JSON keys by default to not trash HTTP caches due to different hash seeds between different workers.
- Added `appcontext_pushed` and `appcontext_popped` signals.
- The builtin run method now takes the `SERVER_NAME` into account when picking the default port to run on.
- Added `flask.request.get_json()` as a replacement for the old `flask.request.json` property.

Version 0.9

Released 2012-07-01, codename Campari

- The `flask.Request.on_json_loading_failed()` now returns a JSON formatted response by default.
- The `flask.url_for()` function now can generate anchors to the generated links.

- The `flask.url_for()` function now can also explicitly generate URL rules specific to a given HTTP method.
- Logger now only returns the debug log setting if it was not set explicitly.
- Unregister a circular dependency between the WSGI environment and the request object when shutting down the request. This means that environ `werkzeug.request` will be `None` after the response was returned to the WSGI server but has the advantage that the garbage collector is not needed on CPython to tear down the request unless the user created circular dependencies themselves.
- Session is now stored after callbacks so that if the session payload is stored in the session you can still modify it in an after request callback.
- The `flask.Flask` class will avoid importing the provided import name if it can (the required first parameter), to benefit tools which build Flask instances programmatically. The Flask class will fall back to using import on systems with custom module hooks, e.g. Google App Engine, or when the import name is inside a zip archive (usually a .egg) prior to Python 2.7.
- Blueprints now have a decorator to add custom template filters application wide, `flask.Blueprint.app_template_filter()`.
- The Flask and Blueprint classes now have a non-decorator method for adding custom template filters application wide, `flask.Flask.add_template_filter()` and `flask.Blueprint.add_app_template_filter()`.
- The `flask.get_flashed_messages()` function now allows rendering flashed message categories in separate blocks, through a `category_filter` argument.
- The `flask.Flask.run()` method now accepts `None` for `host` and `port` arguments, using default values when `None`. This allows for calling run using configuration values, e.g. `app.run(app.config.get('MYHOST'), app.config.get('MYPORT'))`, with proper behavior whether or not a config file is provided.
- The `flask.render_template()` method now accepts either an iterable of template names or a single template name. Previously, it only accepted a single template name. On an iterable, the first template found is rendered.
- Added `flask.Flask.app_context()` which works very similar to the request context but only provides access to the current application. This also adds support for URL generation without an active request context.
- View functions can now return a tuple with the first instance being an instance of `flask.Response`. This allows for returning `jsonify(error="error msg"), 400` from a view function.
- `Flask` and `Blueprint` now provide a `get_send_file_max_age()` hook for subclasses to override behavior of serving static files from Flask when using `flask.Flask.send_static_file()` (used for the default static file handler) and `send_file()`. This hook is provided a filename, which for example allows changing cache controls by file extension. The default max-age for `send_file` and `send_static_file` can be configured through a new `SEND_FILE_MAX_AGE_DEFAULT` configuration variable, which is used in the default `get_send_file_max_age` implementation.

- Fixed an assumption in sessions implementation which could break message flashing on sessions implementations which use external storage.
- Changed the behavior of tuple return values from functions. They are no longer arguments to the response object, they now have a defined meaning.
- Added **`flask.Flask.request_globals_class`** to allow a specific class to be used on creation of the **`g`** instance of each request.
- Added **`required_methods`** attribute to view functions to force-add methods on registration.
- Added **`flask.after_this_request()`**.
- Added **`flask.stream_with_context()`** and the ability to push contexts multiple times without producing unexpected behavior.

Version 0.8.1

Released 2012-07-01

- Fixed an issue with the undocumented **`flask.session`** module to not work properly on Python 2.5. It should not be used but did cause some problems for package managers.

Version 0.8

Released 2011-09-29, codename Rakija

- Refactored session support into a session interface so that the implementation of the sessions can be changed without having to override the Flask class.
- Empty session cookies are now deleted properly automatically.
- View functions can now opt out of getting the automatic OPTIONS implementation.
- HTTP exceptions and Bad Request errors can now be trapped so that they show up normally in the traceback.
- Flask in debug mode is now detecting some common problems and tries to warn you about them.
- Flask in debug mode will now complain with an assertion error if a view was attached after the first request was handled. This gives earlier feedback when users forget to import view code ahead of time.
- Added the ability to register callbacks that are only triggered once at the beginning of the first request. (**`Flask.before_first_request()`**)
- Malformed JSON data will now trigger a bad request HTTP exception instead of a value error which usually would result in a 500 internal server error if not handled. This is a backwards incompatible change.
- Applications now not only have a root path where the resources and modules are located but also an instance path which is the designated place to drop files that are modi-

fied at runtime (uploads etc.). Also this is conceptually only instance depending and outside version control so it's the perfect place to put configuration files etc. For more information see [Instance Folders](#).

- Added the `APPLICATION_ROOT` configuration variable.
- Implemented `session_transaction()` to easily modify sessions from the test environment.
- Refactored test client internally. The `APPLICATION_ROOT` configuration variable as well as `SERVER_NAME` are now properly used by the test client as defaults.
- Added [`flask.views.View.decorators`](#) to support simpler decorating of pluggable (class-based) views.
- Fixed an issue where the test client if used with the “with” statement did not trigger the execution of the teardown handlers.
- Added finer control over the session cookie parameters.
- HEAD requests to a method view now automatically dispatch to the `get` method if no handler was implemented.
- Implemented the virtual `flask.ext` package to import extensions from.
- The context preservation on exceptions is now an integral component of Flask itself and no longer of the test client. This cleaned up some internal logic and lowers the odds of runaway request contexts in unittests.
- Fixed the Jinja2 environment's `list_templates` method not returning the correct names when blueprints or modules were involved.

Version 0.7.2

Released 2011-07-06

- Fixed an issue with URL processors not properly working on blueprints.

Version 0.7.1

Released 2011-06-29

- Added missing future import that broke 2.5 compatibility.
- Fixed an infinite redirect issue with blueprints.

Version 0.7

Released 2011-06-28, codename Grappa

- Added [`make_default_options_response\(\)`](#) which can be used by subclasses to alter the default behavior for `OPTIONS` responses.

- Unbound locals now raise a proper `RuntimeError` instead of an `AttributeError`.
- Mimetype guessing and etag support based on file objects is now deprecated for `flask.send_file()` because it was unreliable. Pass filenames instead or attach your own etags and provide a proper mimetype by hand.
- Static file handling for modules now requires the name of the static folder to be supplied explicitly. The previous autodetection was not reliable and caused issues on Google's App Engine. Until 1.0 the old behavior will continue to work but issue dependency warnings.
- Fixed a problem for Flask to run on jython.
- Added a `PROPAGATE_EXCEPTIONS` configuration variable that can be used to flip the setting of exception propagation which previously was linked to `DEBUG` alone and is now linked to either `DEBUG` or `TESTING`.
- Flask no longer internally depends on rules being added through the `add_url_rule` function and can now also accept regular werkzeug rules added to the url map.
- Added an `endpoint` method to the flask application object which allows one to register a callback to an arbitrary endpoint with a decorator.
- Use Last-Modified for static file sending instead of Date which was incorrectly introduced in 0.6.
- Added `create_jinja_loader` to override the loader creation process.
- Implemented a silent flag for `config.from_pyfile`.
- Added `teardown_request` decorator, for functions that should run at the end of a request regardless of whether an exception occurred. Also the behavior for `after_request` was changed. It's now no longer executed when an exception is raised. See [Upgrading to new Teardown Handling](#)
- Implemented `flask.has_request_context()`
- Deprecated `init_jinja_globals`. Override the `create_jinja_environment()` method instead to achieve the same functionality.
- Added `flask.safe_join()`
- The automatic JSON request data unpacking now looks at the charset mimetype parameter.
- Don't modify the session on `flask.get_flashed_messages()` if there are no messages in the session.
- `before_request` handlers are now able to abort requests with errors.
- It is not possible to define user exception handlers. That way you can provide custom error messages from a central hub for certain errors that might occur during request processing (for instance database connection errors, timeouts from remote resources etc.).
- Blueprints can provide blueprint specific error handlers.
- Implemented generic [Pluggable Views](#) (class-based views).

Released 2010-12-31

- Fixed an issue where the default `OPTIONS` response was not exposing all valid methods in the `Allow` header.
- Jinja2 template loading syntax now allows “./” in front of a template load path. Previously this caused issues with module setups.
- Fixed an issue where the subdomain setting for modules was ignored for the static folder.
- Fixed a security problem that allowed clients to download arbitrary files if the host server was a windows based operating system and the client uses backslashes to escape the directory the files where exposed from.

Version 0.6

Released 2010-07-27, codename Whisky

- After request functions are now called in reverse order of registration.
- `OPTIONS` is now automatically implemented by Flask unless the application explicitly adds ‘`OPTIONS`’ as method to the URL rule. In this case no automatic `OPTIONS` handling kicks in.
- Static rules are now even in place if there is no static folder for the module. This was implemented to aid GAE which will remove the static folder if it’s part of a mapping in the `.yaml` file.
- The `config` is now available in the templates as `config`.
- Context processors will no longer override values passed directly to the render function.
- Added the ability to limit the incoming request data with the new `MAX_CONTENT_LENGTH` configuration value.
- The endpoint for the `flask.Module.add_url_rule()` method is now optional to be consistent with the function of the same name on the application object.
- Added a `flask.make_response()` function that simplifies creating response object instances in views.
- Added signalling support based on blinker. This feature is currently optional and supposed to be used by extensions and applications. If you want to use it, make sure to have `blinker` installed.
- Refactored the way URL adapters are created. This process is now fully customizable with the `create_url_adapter()` method.
- Modules can now register for a subdomain instead of just an URL prefix. This makes it possible to bind a whole module to a configurable subdomain.

 v: 1.1.x ▼

Version 0.5.2

Released 2010-07-15

- Fixed another issue with loading templates from directories when modules were used.

Version 0.5.1

Released 2010-07-06

- Fixes an issue with template loading from directories when modules were used.

Version 0.5

Released 2010-07-06, codename Calvados

- Fixed a bug with subdomains that was caused by the inability to specify the server name. The server name can now be set with the `SERVER_NAME` config key. This key is now also used to set the session cookie cross-subdomain wide.
- Autoescaping is no longer active for all templates. Instead it is only active for `.html`, `.htm`, `.xml` and `.xhtml`. Inside templates this behavior can be changed with the `autoescape` tag.
- Refactored Flask internally. It now consists of more than a single file.
- `flask.send_file()` now emits etags and has the ability to do conditional responses builtin.
- (temporarily) dropped support for zipped applications. This was a rarely used feature and led to some confusing behavior.
- Added support for per-package template and static-file directories.
- Removed support for `create_jinja_loader` which is no longer used in 0.5 due to the improved module support.
- Added a helper function to expose files from any directory.

Version 0.4

Released 2010-06-18, codename Rakia

- Added the ability to register application wide error handlers from modules.
- `after_request()` handlers are now also invoked if the request dies with an exception and an error handling page kicks in.
- Test client has not the ability to preserve the request context for a little longer. This can also be used to trigger custom requests that do not pop the request stack for testing.
- Because the Python standard library caches loggers, the name of the logger is configurable now to better support unittests.

- Added `TESTING` switch that can activate unittesting helpers.
- The logger switches to `DEBUG` mode now if debug is enabled.

Version 0.3.1

Released 2010-05-28

- Fixed a error reporting bug with `flask.Config.from_envvar()`
- Removed some unused code from flask
- Release does no longer include development leftover files (.git folder for themes, built documentation in zip and pdf file and some .pyc files)

Version 0.3

Released 2010-05-28, codename Schnaps

- Added support for categories for flashed messages.
- The application now configures a `logging.Handler` and will log request handling exceptions to that logger when not in debug mode. This makes it possible to receive mails on server errors for example.
- Added support for context binding that does not require the use of the with statement for playing in the console.
- The request context is now available within the with statement making it possible to further push the request context or pop it.
- Added support for configurations.

Version 0.2

Released 2010-05-12, codename J?germeister

- Various bugfixes
- Integrated JSON support
- Added `get_template_attribute()` helper function.
- `add_url_rule()` can now also register a view function.
- Refactored internal request dispatching.
- Server listens on 127.0.0.1 by default now to fix issues with chrome.
- Added external URL support.
- Added support for `send_file()`
- Module support and internal request handling refactoring to better support applications.
- Sessions can be set to be permanent now on a per-session basis.

- Better error reporting on missing secret keys.
- Added support for Google Appengine.

Version 0.1

Released 2010-04-16

- First public preview release.