Becoming Big

Here are your options when growing your codebase or scaling your application.

Read the Source. ¶

Flask started in part to demonstrate how to build your own framework on top of existing well-used tools Werkzeug (WSGI) and Jinja (templating), and as it developed, it became useful to a wide audience. As you grow your codebase, don't just use Flask – understand it. Read the source. Flask's code is written to be read; its documentation is published so you can use its internal APIs. Flask sticks to documented APIs in upstream libraries, and documents its internal utilities so that you can find the hook points needed for your project.

Hook. Extend.

The API docs are full of available overrides, hook points, and Signals. You can provide custom classes for things like the request and response objects. Dig deeper on the APIs you use, and look for the customizations which are available out of the box in a Flask release. Look for ways in which your project can be refactored into a collection of utilities and Flask extensions. Explore the many Extensions in the community, and look for patterns to build your own extensions if you do not find the tools you need.

Subclass.

The `Flask` class has many methods designed for subclassing. You can quickly add or customize behavior by subclassing `Flask` (see the linked method docs) and using that subclass wherever you instantiate an application class. This works well with Application Factories. See Subclassing Flask for an example.

Wrap with middleware.

The Application Dispatching chapter shows in detail how to apply middleware. You can introduce WSGI middleware to wrap your Flask instances and introduce fixes and changes at the layer between your Flask application and your HTTP server. Werkzeug includes several middlewares.

Fork.

If none of the above options work, fork Flask. The majority of code of Flask is within Werkzeug and Jinja2. These libraries do the majority of the work. Flask is just the paste that glues those together. For every project there is the point where the underlying framework gets in the way (due to assumptions the original developers had). This is natural because if this would not be the case, the framework would be a very complex system to begin with which causes a steep learning curve and a lot of user frustration.

This is not unique to Flask. Many people use patched and modified versions of their framework to counter shortcomings. This idea is also reflected in the license of Flask. You don't have to contribute any changes back if you decide to modify the framework.

The downside of forking is of course that Flask extensions will most likely break because the new framework has a different import name. Furthermore integrating upstream changes can be a complex process, depending on the number of changes. Because of that, forking should be the very last resort.

Scale like a pro.

For many web applications the complexity of the code is less an issue than the scaling for the number of users or data entries expected. Flask by itself is only limited in terms of scaling by your application code, the data store you want to use and the Python implementation and webserver you are running on.

Scaling well means for example that if you double the amount of servers you get about twice the performance. Scaling bad means that if you add a new server the application won't perform any better or would not even support a second server.

There is only one limiting factor regarding scaling in Flask which are the context local proxies. They depend on context which in Flask is defined as being either a thread, process or greenlet. If your server uses some kind of concurrency that is not based on threads or greenlets, Flask will no longer be able to support these global proxies. However the majority of servers are using either threads, greenlets or separate processes to achieve concurrency which are all methods well supported by the underlying Werkzeug library.

Discuss with the community.

The Flask developers keep the framework accessible to users with codebases big and small. If you find an obstacle in your way, caused by Flask, don't hesitate to contact the developers on the mailing list or Discord server. The best way for the Flask and Flask extension developers to improve the tools for larger applications is getting feedback from us