Open-Source Report

Flask

An Overview of the Flask Web Framework

Student Group:

Jesse Clapper, Joseph Naro, Matthew Wiewiorski

Teacher:

Jesse Hartloff

Course:

CSE312

# What is Flask?

Flask is an open source web framework meant to provide its users with the tools and libraries necessary to create a variety of web applications. It is a micro-framework with minimal reliance on any external libraries, needing only the Werkzeug utility library and the Jinja2 template engine to function; and making it a light and economical option for web development.

## What does this do for us?

For our purposes we only require most basic usage of Flask in order to handle webserver tasks such as serving files. Flask automatically creates a WSGI webserver that will function as the central object of the webserver when run, with its own functions, templates, and more. There’s a lot of objects Flask automatically manages and creates for its own purposes but for our project we only really worry about manipulating the request object and other basic file serving parts of it. For phase 2 of the project the only required functions we use from Flask are simply the url\_for and redirect functions.

Sequentially our application starts by calling on Flask to generate it’s WSGI application according to specifications. This will not actually open any sockets or do any network tasks yet; this just generates the application. We then proceed to define the homepage for this application as index.html (using Flask functions described below) and at that point we run the application Flask returned.

Once Flask runs the WSGI application it automatically performs webserver tasks, starting with opening a TCP socket on port 5000 (the default value for Flask) and listening for client-side requests. It then takes these requests, parses them, and (if capable) returns an appropriate HTTP response to the client through the same socket connection. A breakdown of all of this code is at <https://github.com/pallets/werkzeug/blob/master/src/werkzeug/serving.py> but for a deeper breakdown:

* After the initialization of the socket on port 5000, application (self) listens for a HTTP client request
* Once a request is logged the function handle\_one\_request extracts the entire request into a variable called self.rawrequestline using the readline function
* If the self.rawrequestline variable is filled the function parse\_request is then called and returns self.run\_wsgi
* run\_wsgi writes a premade header in bytes to a file and then uses key value pairs generated as it parses the request to assemble its HTTP response
* Once completed the function runs end\_headers and terminates the control flow around sending the request back.
* The application will resume waiting for another HTTP request just as it had before sending the response.

## Flask Functions Outside of WSGI Used

## url\_for

Simply generates a URL for a given endpoint. We use this to return a URL to our servers desired file in order to put that file into the Flask request object and push it to the client.

## redirect

Returns a response object based on the given URL and code which, when called on, sends the client that HTTP response.

## Licensing

Flask operates under a source license that dictates that all content within the Flask repository or source distribution must be accompanied with the copyright “Copyright 2010 Pallets” and are not used to advertise personal projects. This doesn’t really effect our code in anyways be we do not redistributing or modifying the source code in any way.