# Werkzeug

Werkzeug is a comprehensive WSGI web application library. It began as a simple collection of various utilities for WSGI applications and has become one of the most advanced WSGI utility libraries. It includes:

* An interactive debugger that allows inspecting stack traces and source code in the browser with an interactive interpreter for any frame in the stack.
* A full-featured request object with objects to interact with headers, query args, form data, files, and cookies.
* A response object that can wrap other WSGI applications and handle streaming data.
* A routing system for matching URLs to endpoints and generating URLs for endpoints, with an extensible system for capturing variables from URLs.
* HTTP utilities to handle entity tags, cache control, dates, user agents, cookies, files, and more.
* A threaded WSGI server for use while developing applications locally.
* A test client for simulating HTTP requests during testing without requiring running a server.

Werkzeug doesn’t enforce any dependencies. It is up to the developer to choose a template engine, database adapter, and even how to handle requests. It can be used to build all sorts of end user applications such as blogs, wikis, or bulletin boards. Flask wraps Werkzeug, using it to handle the details of WSGI while providing more structure and patterns for defining powerful applications.

## Fundamentals

Werkzeug, as mentioned, is a utility library for WSGI. WSGI itself is a protocol or convention that ensures that the web application can speak with the webserver and more importantly that web applications work nicely together.

A WSGI application is something one can call and pass an environ dict and a start\_response callable. The environ contains all incoming information, the start\_response function can be used to indicate the start of the response. With Werkzeug there is no need to deal directly with either as request and response objects are provided to work with them. The request data takes the environ object and allows you to access the data from that environ in a nice manner. The response object is a WSGI application in itself and provides a much nicer way to create responses.

## Routing

Routing is the process of matching and parsing the URL to something we can use. Werkzeug provides a flexible integrated routing system which we can use for that. The way it works is that you create a Map instance and add a bunch of Rule objects. Each rule has a pattern it will try to match the URL against and an “endpoint”. The endpoint is typically a string and can be used to uniquely identify the URL. We could also use this to automatically reverse the URL, but that’s not what we will do in this tutorial.

# Bibliography

[1] Werkzeug Documentation (<https://werkzeug.palletsprojects.com/en/2.0.x/>)