Flask is a lightweight Python framework for web applications that provides the basics for URL routing and page rendering.

Flask is called a "micro" framework because it doesn't directly provide features like form validation, database abstraction, authentication, and so on. Such features are instead provided by special Python packages called Flask extensions. The extensions integrate seamlessly with Flask so that they appear as if they were part of Flask itself. For example, Flask doesn't provide a page template engine, but installing Flask includes the Jinja templating engine by default. For convenience, we typically speak of these defaults as part of Flask.

**What is Flask?**

Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by **Armin Ronacher** who leads an international group of python enthusiasts (POCCO). It is based on WSGI toolkit and jinja2 template engine. Flask is considered as a micro framework.

**What is WSGI?**

It is an acronym for web server gateway interface which is a standard for python web application development. It is considered as the specification for the universal interface between the web server and web application. between JDK, JRE, and JVM

**What is Jinja2?**

Jinja2 is a web template engine which combines a template with a certain data source to render the dynamic web pages.

### **Advantages of Python Flask**

1. It is a lightweight and modular design  
2. Contains a built-in development server and a fast debugger.  
3. Provides integrated unit testing support  
4. RESTful request dispatching.  
5. Jinja2 Template.  
6. Provides support for secure cookies.

# **Flask App routing**

App routing is used to map the specific URL with the associated function that is intended to perform some task

In our first application, the URL ('/') is associated with the home function that returns a particular string displayed on the web page.

In other words, we can say that if we visit the particular URL mapped to some particular function, the output of that function is rendered on the browser's screen.

## The add\_url\_rule() function

There is one more approach to perform routing for the flask web application that can be done by using the add\_url() function of the Flask class. The syntax to use this function is given below.

1. add\_url\_rule(<url rule>, <endpoint>, <view function>)

This function is mainly used in the case if the view function is not given and we need to connect a view function to an endpoint externally by using this function.

# **Flask URL Building**

The url\_for() function is used to build a URL to the specific function dynamically. The first argument is the name of the specified function, and then we can pass any number of keyword argument corresponding to the variable part of the URL.

This function is useful in the sense that we can avoid hard-coding the URLs into the templates by dynamically building them using this function.

# **Flask HTTP methods**

HTTP is the hypertext transfer protocol which is considered as the foundation of the data transfer in the world wide web. All web frameworks including flask need to provide several HTTP methods for data communication.

The methods are given in the following table.

|  |  |  |
| --- | --- | --- |
| **SN** | **Method** | **Description** |
| 1 | GET | It is the most common method which can be used to send data in the unencrypted form to the server. |
| 2 | HEAD | It is similar to the GET but used without the response body. |
| 3 | POST | It is used to send the form data to the server. The server does not cache the data transmitted using the post method. |
| 4 | PUT | It is used to replace all the current representation of the target resource with the uploaded content. |
| 5 | DELETE | It is used to delete all the current representation of the target resource specified in the URL. |

We can specify which HTTP method to be used to handle the requests in the route() function of the Flask class. By default, the requests are handled by the GET() method.