

Flask Tutorial

Flask is a python based **web application framework**. Web Application Framework or simply Web Framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management etc.

Environment Setup

1. Python 2.6 or higher is required for installation of Flask.

2. Install virtualenv for development environment

virtualenv is a virtual Python environment builder. It helps a user to create multiple Python environments side-by-side. Thereby, it can avoid compatibility issues between the different versions of the libraries.

Note : if you're using Pycharm, you don't have to worry about this.

3. Install Flask in this environment

pip install Flask

Hello World program

```
from flask import Flask
app = Flask(__name__)

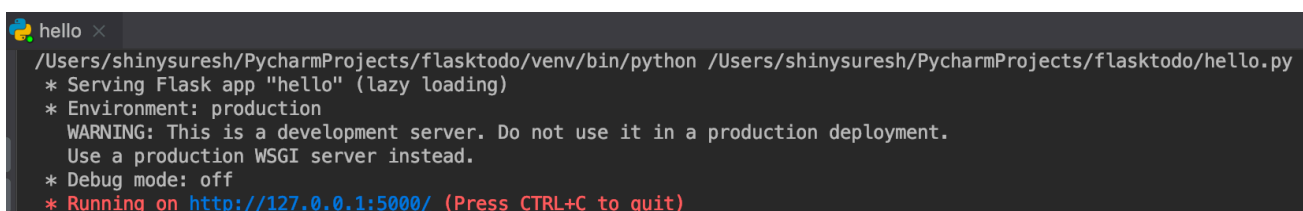
@app.route('/')
def hello():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

1. Import the Flask class.
2. Create an instance of this class. The first argument is the name of the application's module or package. (If you are using a single module, __name__ is always the correct value. If you however are using a package, it's usually recommended to hardcode the name of your package)
3. Use the route() decorator to tell Flask what URL should trigger our function.
4. Finally the run() method of Flask class runs the application on the local development server.

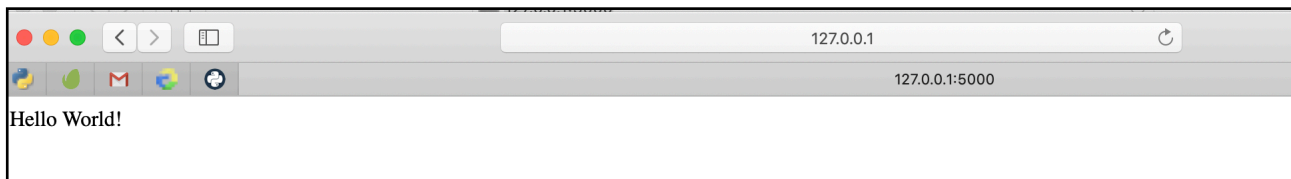
Save your program as hello.py or something similar. Make sure to not call your application flask.py because this would conflict with Flask itself.

Run the application by running the 'hello.py' file.



```
hello x
/Users/shinysuresh/PycharmProjects/flasktodo/venv/bin/python /Users/shinysuresh/PycharmProjects/flasktodo/hello.py
* Serving Flask app "hello" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Open the URL <http://127.0.0.1:5000/> in your browser and 'Hello World!' Should appear.



Debug mode

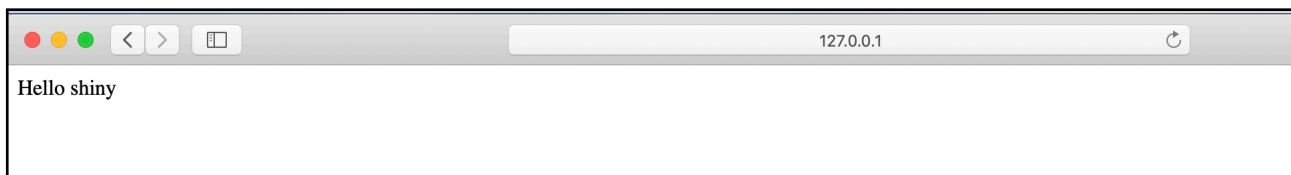
A Flask application is started by calling the `run()` method. However, while the application is under development, it should be restarted manually for each change in the code. To avoid this inconvenience, enable debug support. The server will then reload itself if the code changes. It will also provide a useful debugger to track the errors if any, in the application.

In order to enable the debug mode, modify the line `app.run()` to `app.run(debug=True)`.

Hello With Your Name

```
@app.route("/<name>")
def hello_name(name):
    return "Hello " + name
```

Open the URL <http://127.0.0.1:5000/shiny> in your browser.



URL Building

To build a URL to a specific function, use the `url_for()` function. The function accepts the name of a function as first argument, and one or more keyword arguments, each corresponding to the variable part of URL.

Example:

```
from flask import Flask, redirect, url_for
app = Flask(__name__)

@app.route('/admin')
def hello_admin():
    return 'Hello Admin'

@app.route('/guest/<guest>')
def hello_guest(guest):
```

```
return 'Hello %s as Guest' % guest

@app.route('/user/<name>')
def hello_user(name):
    if name == 'admin':
        return redirect(url_for('hello_admin'))
    else:
        return redirect(url_for('hello_guest', guest = name))

if __name__ == '__main__':
    app.run(debug = True)
```

RESTful Web Services

A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Web services based on REST Architecture are known as RESTful web services. REST stands for REpresentational State Transfer. These webservices uses HTTP methods to implement the concept of REST architecture. RESTful web services uses the concept of resources. Resources are represented by [URIs](#)(Uniform Resource Identifier). The clients send requests to these URIs using the methods defined by the HTTP protocol, and possibly as a result of that the state of the affected resource changes.

HTTP Methods

Web applications use different HTTP methods when accessing URLs.

GET - to get data from a resource (read only access)

PUT - to update data at a resource

POST - to create data at a resource

DELETE - to delete data at a resource

PATCH - to partially update data at a resource

By default, the Flask route responds to the GET requests. However, this preference can be altered by providing methods argument to route() decorator.

```
@app.route('/add', methods=['POST'])
def add():
    add_todo_item(text=request.form['todoitem'])
    return redirect(url_for('index'))
```

Static Files

Dynamic web applications also need static files. That's usually where the CSS and JavaScript files are coming from. Create a folder called static in your package or next to your module and it will be available at /static on the application.

To generate URLs for static files, use the special 'static' endpoint name:

`url_for('static', filename='style.css')`

The file has to be stored on the filesystem as static/style.css.

Templates

Templates are files that contain static data as well as placeholders for dynamic data. A template is rendered with specific data to produce a final document. In your application, you will use templates to render [HTML](#) which will display in the user's browser. Flask configures the Jinja2 template engine automatically.

Jinja looks and behaves mostly like Python. Special delimiters are used to distinguish Jinja syntax from the static data in the template. Anything between `{{` and `}}` is an expression that will be output to the final document. `{%` and `%}` denotes a control flow statement like 'if' and 'for'. Unlike Python, blocks are denoted by start and end tags rather than indentation.

To render a template you can use the `render_template()` method. All you have to do is provide the name of the template and the variables you want to pass to the template engine as keyword arguments.

Here's a simple example of how to render a template:

```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)
```

Flask will look for templates in the templates folder. So if your application is a module, this folder is next to that module, if it's a package it's actually inside your package:

```
case 1: a module:
/application.py
/templates
  /hello.html

case 2: a package:
/application
  /__init__.py
  /templates
    /hello.html
```

References

<https://flask.palletsprojects.com/>
<https://flask.palletsprojects.com/en/1.0.x/quickstart/>
<http://flask.pocoo.org/docs/1.0/quickstart/>
https://www.tutorialspoint.com/flask/flask_application
<https://www.programiz.com/python-programming/decorator>