



Introduction to Flask

- Aishwarya Gulve

Overview of the Flask

What is Web Framework?

Web application framework or web framework is a collection of packages or modules which allow developer to write web applications without having to bother about low-level details such as protocols, thread management etc.

What is Flask?

Flask is a web framework that provides libraries to build lightweight web applications in python. It is based on Werkzeug WSGI toolkit and jinja2 template engine

What is Werkzeug?

It is a **WSGI toolkit**, which implements **requests, response objects, and other utility functions**. This enables building a web framework on top of it.

What is jinja2?

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

Flask Environment Setup

Flask Environment Setup

Prerequisite: We need to have python 2.7 or higher for installation of flask

Install virtual environment: It considered as the virtual python environment builder. It is used to create the multiple python virtual environment side by side

Install by using following command:

```
Pip install virtualenv
```


Flask Environment Setup

Create new virtual environment: Once it is installed, create the new virtual environment into a folder

```
mkdir mynewproj  
cd mynewproj  
virtualenv venv
```

Flask Environment Setup

Activate the corresponding environment, use the following command on linux operating system:

```
venv/bin/activate
```

Use the following command on windows operating system:

```
venv\scripts\activate
```

Flask Environment Setup

Now we can install flask in this environment. Use following command to install flask:

```
Pip install flask
```

Flask Application

First Flask Application

Now, we will build our first python website using the Flask framework. In this tutorial we are using the **jupyter notebook**.

In order to check flask installation, write following lines of code in the jupyter notebook.

```
# import flask
from flask import Flask

app = Flask(__name__) # creating the flask class object

@app.route('/') # decorator
def home():
    return "Hello, we are creating our first flask website";

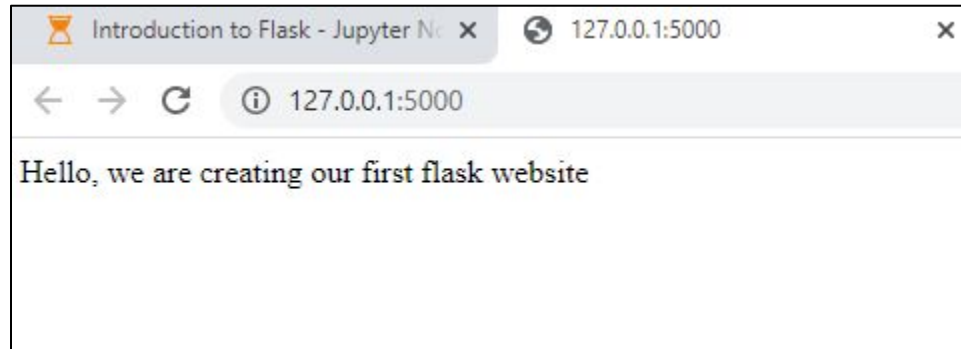
# run the application
if __name__ == '__main__':
    app.run(debug=False) # use debug = False for jupyter notebook
```

* Running ▶ http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [09/Aug/2020 15:28:44] "GET / HTTP/1.1" 200 -

Open this URL in the browser. The message will be displayed on it.

First Flask Application

Finally, we run the flask application on the local development server



Understand the Code

```
# import flask
from flask import Flask

app = Flask(__name__) # creating the flask class object

@app.route('/') # decorator
def home():
    return "Hello, we are creating our first flask website";

# run the application
if __name__ == '__main__':
    app.run(debug=False) # use debug = False for jupyter notebook

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [09/Aug/2020 15:28:44] "GET / HTTP/1.1" 200 -
```

Pass the name of the current module. i.e. `__name__` as the argument into the flask constructor

`route()` : It is a decorator, which tells the application which URL should call the associated function. In this example, `'/'` URL is bound with `home()` function

`run()` method: It is used to run the flask application on the local development server.

Flask - Routing

What is Flask App Routing?

- It is used to map specific URL with the associated function that is to perform some task. It is used to access some particular page in the web application
- In the above created application, the URL('/') is associated with the home function that returns a particular string displayed on the web page

Flask App Routing

- **So, what is flask routing?**

The routes are the different URLs that the application implements. The handlers for the application routes are written as Python functions

- The Python functions are mapped to one or more URLs so that flask knows what logic to execute when a client requests a given URL

Example:1

```
# import flask
from flask import Flask

app = Flask(__name__) # creating the flask class object

@app.route('/') # decorator
@app.route('/FlaskTutorial') # decorator
def FlaskTutorial(): # python function
    return "Hello, Welcome to our Flask Tutorial";

# run the application
if __name__ == "__main__":
    app.run(debug = False)
```

```
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [10/Aug/2020 13:32:39] "GET /FlaskTutorial HTTP/1.1" 200 -
127.0.0.1 - - [10/Aug/2020 13:32:39] "GET / HTTP/1.1" 200 -
```

- In this example, there are two decorators, which associate the URLs / and /FlaskTutorial to this function
- This means that when a web browser requests either of these two URLs, Flask is going to call on this function and pass the return value of it back to the browser as a response

Example:1

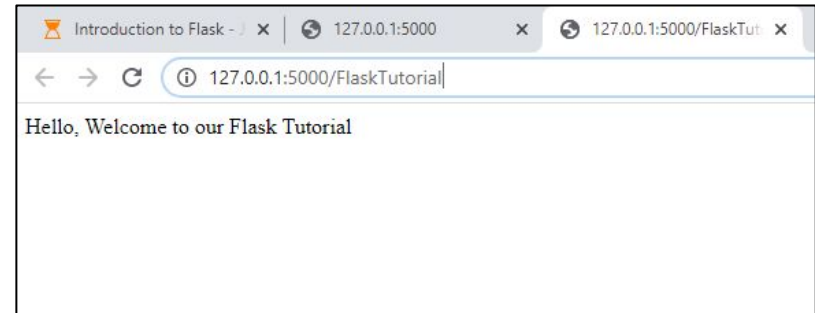
```
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [10/Aug/2020 13:32:39] "GET /FlaskTutorial HTTP/1.1" 200 -
127.0.0.1 - - [10/Aug/2020 13:32:39] "GET / HTTP/1.1" 200 -
```

- The output from the flask run indicates that the server is running on IP address 127.0.0.1
- Now open your web browser and enter the following URL in the address field:

<http://127.0.0.1:5000/>

Alternatively we use other URL:

<http://127.0.0.1:5000/FlaskTutorial>



Example:2

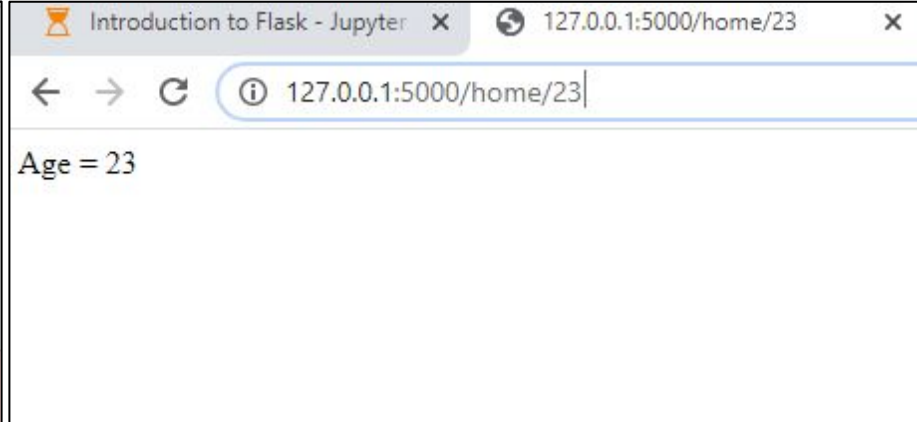
- We can use converter in the URL to map the specified variable to the particular data type
- In this example, we are providing the integer variable age

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

@app.route('/')
@app.route('/home/<int:age>')
def home(age):
    return "Age = %d"%age;

if __name__ == "__main__":
    app.run(debug = False)
```

```
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [10/Aug/2020 16:31:19] "GET /home HTTP/1.1" 404 -
```



add_url_rule() Function

- We can use add_url_rule() to perform routing for the flask web application

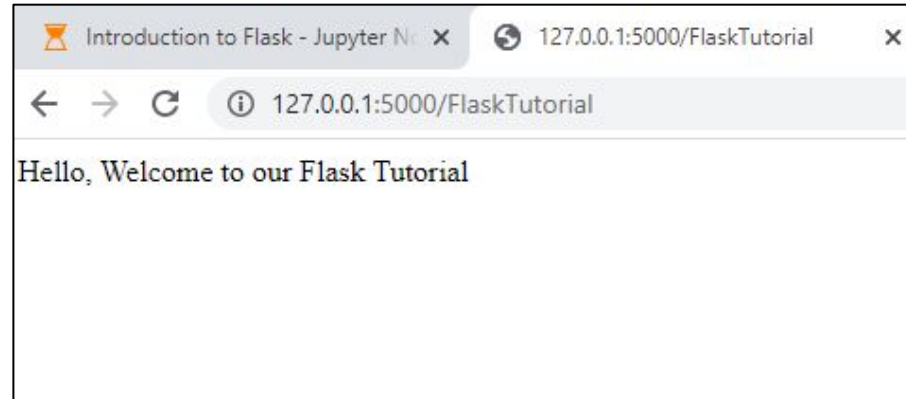
```
from flask import Flask
app = Flask(__name__)

def FlaskTutorial():
    return "Hello, Welcome to our Flask Tutorial";

app.add_url_rule("/FlaskTutorial", "FlaskTutorial", FlaskTutorial)

if __name__ == "__main__":
    app.run(debug = False)

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



URL Building

URL Building

- We can use `url_for()` function to build URL in flask
- The function is accepted as its first argument and any number of keyword arguments
- We can avoid hard-coded URLs by changing URLs in one shot instead of hard coding

URL Building

```
from flask import Flask, url_for

app = Flask(__name__)

@app.route('/admin')
def welcome_admin():
    return 'Welcome Admin'

@app.route('/guest/<guest>')
def welcome_guest(guest):
    return 'Welcome %s' % guest

@app.route('/user/<username>')
def student(username):
    return '{} Student'.format(username)

@app.route('/users/<name>')
def welcome_user(name):
    if name == 'admin':
        return redirect(url_for('welcome_admin'))
    else:
        return redirect(url_for('welcome_guest', guest = name))

if __name__ == "__main__":
    app.run(debug = False)
```

* Running on <http://127.0.0.1:5000/> (Press CTRL+C to quit)

Introduction to Flask - Jupyter Notebook x 127.0.0.1:5000/guest/Aishwarya x

← → ↻ ⓘ 127.0.0.1:5000/guest/Aishwarya

Welcome Aishwarya

Introduction to Flask - Jupyter Notebook x 127.0.0.1:5000/admin

← → ↻ ⓘ 127.0.0.1:5000/admin

Welcome Admin

URL Building

- We can use `url_for()` function to build URL in flask
- The function is accepted as its first argument and any number of keyword arguments
- We can avoid hard-coded URLs by changing URLs in one shot instead of hard coding

Flask HTTP Methods

HTTP Methods

- HTTP is stands for **hypertext transfer protocol**
- It is considered as the foundation of the data transfer in the world wide web
- All the web frameworks need to provide several HTTP methods for data communication
- Specify the HTTP method to handle the request in the route() function of the flask class
- By default, the requests are handled by the GET() method

HTTP Methods

Method	Description
GET	It sends data in unencrypted form to the server
HEAD	It is similar to the get method, but without response body
POST	It is used to send HTML data to server. The server does not cache the data transmitted using the post method
PUT	It replaces all current representations of the target resource with the uploaded content
DELETE	It removes all current representations of the target resource given by a URL

POST Method

- Create an HTML form and use the post method to send data to a URL
- Save the below script as login.html

```
<html>

  <body>

    <form action = "http://localhost:5000/login" method = "post">

      <table>

        <tr><td>Name</td>

        <td><input type ="text" name ="uname"></td></tr>

        <tr><td>Password</td>

        <td><input type ="password" name ="pass"></td></tr>

        <tr><td><input type = "submit"></td></tr>

      </table>

    </form>

  </body>

</html>
```

POST Method

- Now, Write the following code into the jupyter notebook

```
from flask import Flask, redirect, url_for, request

app = Flask(__name__)

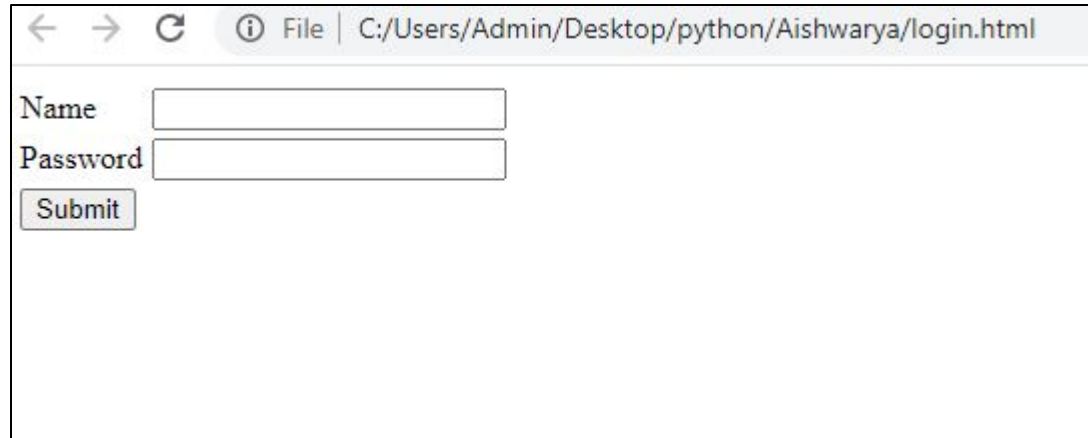
@app.route('/login', methods = ['POST'])
def login():
    username=request.form['uname']
    passwd=request.form['pass']
    if username=="Aishwarya" and passwd=="XYZ":
        return "Welcome %s" %username

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

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

POST Method

- Open login.html on web browser



A screenshot of a web browser window displaying a login form. The browser's address bar shows the file path `C:/Users/Admin/Desktop/python/Aishwarya/login.html`. The form contains two input fields: one for 'Name' and one for 'Password'. Below these fields is a 'Submit' button.

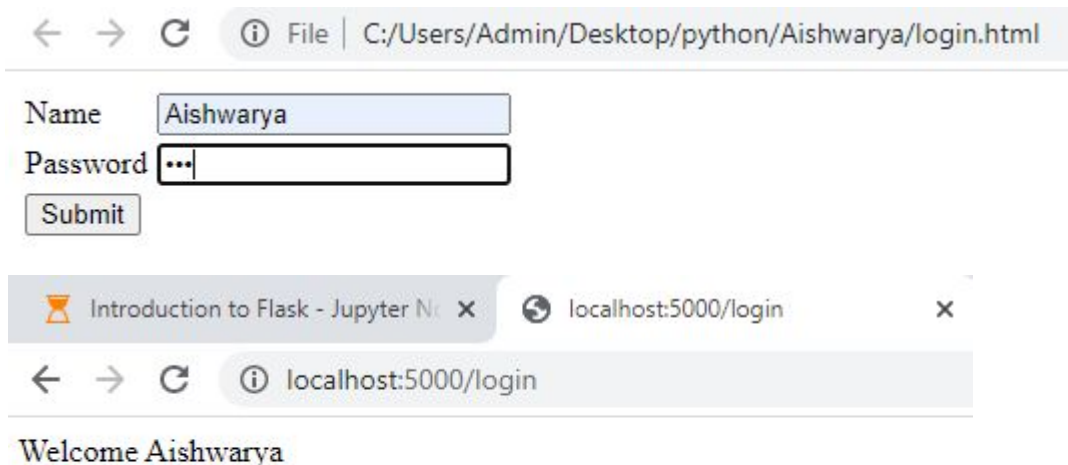
← → ↻ ⓘ File | C:/Users/Admin/Desktop/python/Aishwarya/login.html

Name

Password

POST Method

- Give the required input and click submit



The image shows a sequence of browser screenshots. The top screenshot displays a file explorer view of a login.html file. The form contains a 'Name' field with 'Aishwarya', a 'Password' field with masked characters, and a 'Submit' button. The bottom screenshot shows the browser's address bar at localhost:5000/login, with the page content displaying 'Welcome Aishwarya'.

← → ↻ File | C:/Users/Admin/Desktop/python/Aishwarya/login.html

Name

Password

Introduction to Flask - Jupyter Notebook x localhost:5000/login x

← → ↻ localhost:5000/login

Welcome Aishwarya

GET Method

- Consider the same example for get method
- Save the below script as login.html

```
<html>

  <body>

    <form action = "http://localhost:5000/login" method = "get">

      <table>

        <tr><td>Name</td>

        <td><input type ="text" name ="uname"></td></tr>

        <tr><td>Password</td>

        <td><input type ="password" name ="pass"></td></tr>

        <tr><td><input type = "submit"></td></tr>

      </table>

    </form>

  </body>

</html>
```

GET Method

- Now, Write the following code into the jupyter notebook

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

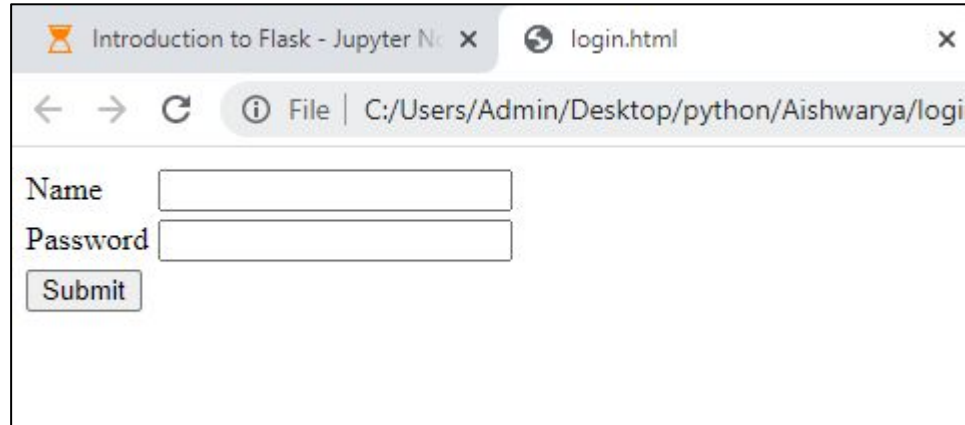
@app.route('/login', methods = ['GET'])
def login():
    uname=request.args.get('uname')
    passwd=request.args.get('pass')
    if uname=="Aishwarya" and passwd=="XYZ":
        return "Welcome %s" %uname

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

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

GET Method

- Open login.html on web browser



A screenshot of a web browser window. The address bar shows the file path `C:/Users/Admin/Desktop/python/Aishwarya/login.html`. The page content includes a form with two input fields: "Name" and "Password", and a "Submit" button.

Introduction to Flask - Jupyter Notebook x login.html x

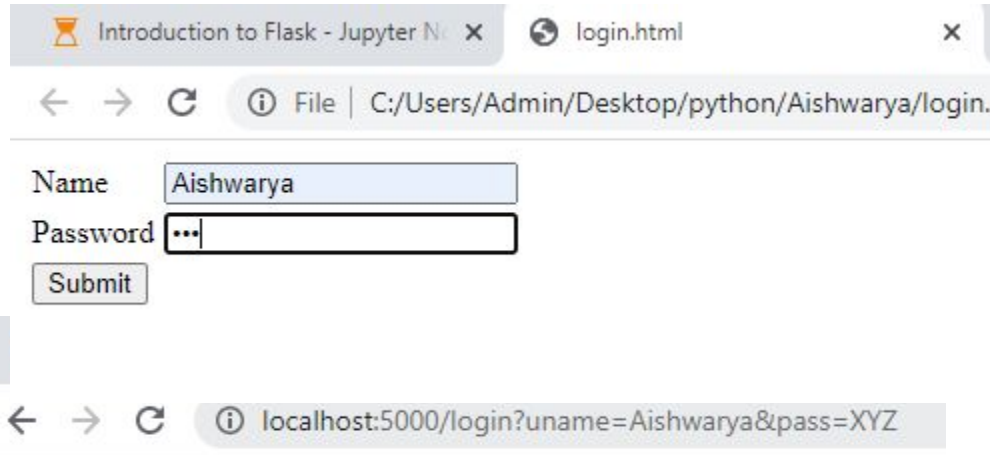
← → ↻ ⓘ File | C:/Users/Admin/Desktop/python/Aishwarya/login.html

Name

Password

GET Method

- Give the required input and click submit



The image shows a web browser window with two tabs. The first tab is titled 'Introduction to Flask - Jupyter N...' and the second tab is titled 'login.html'. The address bar shows the file path 'C:/Users/Admin/Desktop/python/Aishwarya/login.'. Below the address bar, there is a login form with two input fields: 'Name' and 'Password'. The 'Name' field contains the text 'Aishwarya'. The 'Password' field contains three dots '...'. Below the input fields is a 'Submit' button. The browser window is then shown again, but the address bar now displays 'localhost:5000/login?uname=Aishwarya&pass=XYZ', indicating that the form has been submitted and the data is being sent via a GET request.

Introduction to Flask - Jupyter N x login.html x

← → ↻ ⓘ File | C:/Users/Admin/Desktop/python/Aishwarya/login.

Name

Password

← → ↻ ⓘ localhost:5000/login?uname=Aishwarya&pass=XYZ

Welcome Aishwarya

GET Method

- In the above image, we can see the URL which contains the data sent with the request to the server
- This is the important difference between the GET requests and the POST requests as the data sent to the server is not shown in the URL on the browser in the POST requests

Flask Templates

Example

```
from flask import Flask, redirect, url_for, request

app = Flask(__name__)
@app.route('/')

def text():
    return"<html><body><h1>Hi, welcome to my channel</h1><body><html>"

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

```
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- This example contains a view function, i.e., the `text()` which is associated with the URL `'/'`. Instead of returning a simple plain string as a text, it returns a text with `<h1>` tag attached to it using HTML

Render External HTML Files

- We can render the external HTML file instead of hardcoding the HTML in the view function
- Flask provides us the `render_template()` function which can be used to render the external HTML file to be returned as the response from the view function

Render External HTML Files

- Create an HTML file named as text.html to render an HTML file from the view function
- Create the folder templates inside the application directory and save the HTML templates referenced in the flask script in that directory

```
<html>
<head>
<title>text</title>
</head>
<body>
<h1>Hi, welcome to my channel </h1>
</body>
</html>
```

Render External HTML Files

- Now, Write the following code into the jupyter notebook

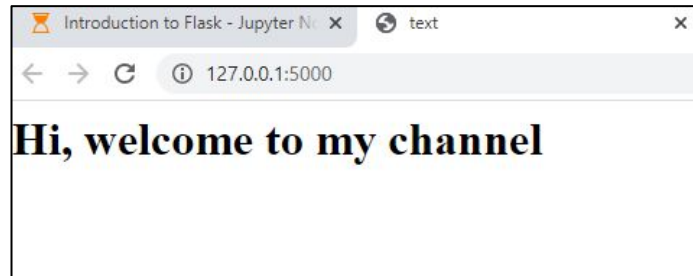
```
from flask import Flask, redirect, url_for, request, render_template

app = Flask(__name__)
@app.route('/')

def text():
    return render_template('text.html')

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

* Running on <http://127.0.0.1:5000/> (Press CTRL+C to quit)



Delimiters

- The jinja2 template engine provides some delimiters which can be used in the HTML to make it capable of dynamic data representation
- The jinja2 template engine provides the following delimiters to escape from the HTML

Method	Description
<code>{%...%}</code>	for statements
<code>{{...}}</code>	for expressions to print to the template output
<code>{#...#}</code>	for the comments that are not included in the template output
<code>#...##</code>	for line statements

Example:1

- Use {{...}} delimiter to show the variable part of the URL in the HTML script

```
<html>
<head>
<title>Message</title>
</head>
<body>
<h1>hi, {{ name }}</h1>
</body>
</html>
```

Example:1

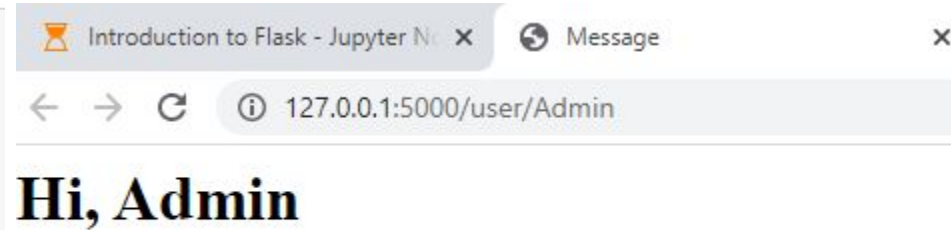
- The variable part of the URL <http://localhost:5000/user/Admin> is shown in the HTML script using the {{name}} placeholder

```
from flask import Flask, redirect, url_for, request, render_template

app = Flask(__name__)
@app.route('/user/<uname>')
def text(uname):
    return render_template('text.html', name=uname)

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

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



Example:2

- In this example, we will print the table of a number specified in the URL

```
<html>
<head>
<title>print table</title>
</head>
<body>
<h2> printing table of {{n}}</h2>
{% for i in range(1,11): %}
    <h3>{{n}} X {{i}} = {{n * i}} </h3>
{% endfor %}
</body>
</html>
```

Example:2

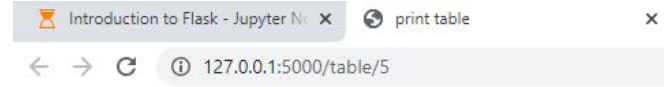
- Write the following code in the jupyter notebook

```
from flask import Flask, redirect, url_for, request, render_template

app = Flask(__name__)
@app.route('/table/<int:num>')
def table(num):
    return render_template('table.html', n=num)

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

* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```



printing table of 5

5 X 1 = 5

5 X 2 = 10

5 X 3 = 15

5 X 4 = 20

5 X 5 = 25

5 X 6 = 30

5 X 7 = 35

5 X 8 = 40

5 X 9 = 45

5 X 10 = 50

THANK YOU

This is the first series of the flask.