Python | Introduction to Web development using Flask

What is Flask?

Flask is an API of Python that allows to build up web-applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django's framework and is also easier to learn because it have less base code to implement a simple web-Application. A Web-Application Framework or Web Framework is the collection of modules and libraries that helps the developer to write applications without writing the low-level codes such as protocols, thread management, etc. Flask is based on WSGI(Web Server Gateway Interface) toolkit and Jinja2 template engine.

Getting Started With Flask:

Python 2.6 or higher is required for installation of the Flask. You can start by import Flask from flask package on any python IDE. For installation on any environment, you can click on the installation link given below.

To test that if the installation is working, check out this code given below.

```
# an object of WSGI application
from flask import Flask
app = Flask(__name__)   # Flask constructor

# A decorator used to tells the application
# which URL is associated function
@app.route('/')
def hello():
    return 'HELLO'

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

'/' URL is bound with `hello()` function. When the home page of web server is opened in browser, the output of this function will be rendered accordingly.

The Flask application is started by calling the `run()` function. The method should be restarted manually for any change in the code. To overcome this, the debug support is enabled so as to track any error.

```
app.debug = True
app.run()
app.run(debug = True)
```

Routing:

Nowadays, the web frameworks provide routing technique so that user can remember the URLs. It is useful to access the web page directly without navigating from the Home page. It is done through the following `route()` decorator, to bind the URL to a function.

```
# decorator to route URL
@app.route('/hello')
```

```
# binding to the function of route
def hello_world():
   return ‘hello world’
```

If a user visits http://localhost:5000/hello URL, the output of the hello_world() function will be rendered in the browser. The `add_url_rule()`function of application object can also be used to bind URL with the function as in above example.

```
def hello_world():
   return ‘hello world’
app.add_url_rule(‘/’, ‘hello’, hello_world)
```

Using Variables in Flask:

The Variables in the flask is used to build a URL dynamically by adding the variable parts to the rule parameter. This variable part is marked as. It is passed as keyword argument. See the example below

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

# routing the decorator function hello_name
@app.route('/hello/')
def hello_name(name):
   return 'Hello %s!' % name

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

Save the above example as hello.py and run from power shell. Next, open the browser and enter the URL http://localhost:5000/hello/GeeksforGeeks.

Output:

```
Hello GeeksforGeeks!
```

In the above example, the parameter of route() decorator contains variable part attached to the URL '/hello' as an argument. Hence, if URL like http://localhost:5000/hello/GeeksforGeeks is entered then 'GeeksforGeeks' will be passed to the hello() function as an argument.

In addition to the default string variable part, other datatypes like int, float and path(for directory separator channel which can take slash) is also used. The URL rules of Flask are based on Werkzeug's routing module. This ensures that the URLs formed are unique and based on precedents laid down by Apache.

Examples:

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

@app.route('/blog/')
def show_blog(postID):
   return 'Blog Number %d' % postID

@app.route('/rev/')
def revision(revNo):
   return 'Revision Number %f' % revNo

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

# say the URL is http://localhost:5000/blog/555
```

Output :

```
Blog Number 555

# Enter this URL in the browser ? http://localhost:5000/rev/1.1
Revision Number: 1.100000
```

Building URL in FLask:

Dynamic Building of the URL for a specific function is done using `url_for()` function. The function accepts the name of the function as first argument, and one or more keyword arguments. See this example

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

@app.route('/admin')  #decorator for route(argument) function
def hello_admin():     #binding to hello_admin call
   return 'Hello Admin'

@app.route('/guest/')
def hello_guest(guest):    #binding to hello_guest call
   return 'Hello %s as Guest' % guest

@app.route('/user/')
def hello_user(name):
```

```
        if name =='admin':  #dynamic binding of URL to function
           return redirect(url_for('hello_admin'))
        else:
           return redirect(url_for('hello_guest', guest = name))

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

To test this, save the above code and run through python shell and then open browser and enter the following URL:-

Input: http://localhost:5000/user/admin
Output: Hello Admin

Input: http://localhost:5000/user/ABC
Output: Hello ABC as Guest

The above code have a function named user(name), accepts the value through input URL. It basically checks that the received argument matches the ‘admin’ argument or not. If it matches, then the function hello_admin() is called else the hello_guest() is called.

Flask support various HTTP protocols for data retrieval from the specified URL, these can be defined as:-

METHOD	DESCRIPTION
GET	This is used to send the data in an without encryption of the form to the server.
HEAD	provides response body to the form
POST	Sends the form data to server. Data received by POST method is not cached by server.
PUT	Replaces current representation of target resource with URL.
DELETE	Deletes the target resource of a given URL

Handling Static Files :

A web application often requires a static file such as javascript or a CSS file to render the display of the web page in browser. Usually, the web server is configured to set them, but during development, these files are served as static folder in your package or next to the module. See the example in JavaScript given below:

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

@app.route("/")
def index():
   return render_template("index.html")

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

The following HTML code:

```
<html>

   <head>
      <script type = "text/javascript"
         src = "{{ url_for('static', filename = 'hello.js') }}" ></script>
   </head>

   <body>
      <input type = "button" onclick = "sayHello()" value = "Say Hello" />
   </body>

</html>
```

The JavaScript file for `hello.js` is :

```
function sayHello() {
   alert("Hello World")
}
```

The above hello.js file will be rendered accordingly to the HTML file.

Object Request of Data from a client's web page is send to the server as a global request object. It is then processed by importing the Flask module. These consist of attributes like Form(containing Key-Value Pair), Args(parsed URL after question mark(?)), Cookies(contain Cookie names and Values), Files(data pertaining to uploaded file) and Method(current request).

Cookies:

A Cookie is a form of text file which is stored on a client's computer, whose purpose is to remember and track data pertaining to client's usage in order to improve the website according to the user's experience and statistic of webpage.

The Request object contains cookie's attribute. It is the dictionary object of all the cookie variables and their corresponding values. It also contains expiry time of itself. In Flask, cookie are set on response object.See the example given below:-

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def index():
   return render_template('index.html')
```

```
<html>
   <body>

      <form action = "/setcookie" method = "POST">
         <p><h3>Enter userID</h3></p>
         <p><input type = 'text' name = 'nm'/></p>
         <p><input type = 'submit' value = 'Login'/></p>
      </form>
```

```
   </body>
</html>



@app.route('/setcookie', methods = ['POST', 'GET'])
def setcookie():
   if request.method == 'POST':
      user = request.form['nm']

   resp = make_response(render_template('cookie.html'))
   resp.set_cookie('userID', user)

   return resp

@app.route('/getcookie')
def getcookie():
   name = request.cookies.get('userID')
   return '<h1>welcome '+name+'</h1>'
```

Run the above application and visit link on Browser http://localhost:5000/

The form is set to '/setcookie' and function set contains a Cookie name userID that will be rendered to another webpage. The 'cookie.html' contains hyperlink to another view function getcookie(), which displays the value in browser.

Sessions in Flask:

In Session, the data is stored on Server. It can be defined as a time interval in which the client logs into a server till the user log out. The data in between them are hold in temporary folder on the Server. Each user is assigned with a specific **Session ID**. The Session object is a dictionary that contains the key-value pair of the variables associated with session. A SECRET_KEY is used to store the encrypted data on the cookie.

For example:

```
Session[key] = value   // stores the session value
Session.pop(key, None)  // releases a session variable
```

Other Important Flask Functions:

`redirect()`: It is used to return the response of an object and redirects the user to another target location with specified status code.

```
Syntax: Flask.redirect(location, statuscode, response)
```

//location is used to redirect to the desired URL

//statuscode sends header value, default 302

//response is used to initiate response.

abort: It is used to handle the error in the code.

```
Syntax:  Flask.abort(code)
```

The code parameter can take the following values to handle the error accordingly:

- **400** – For Bad Request
- **401** – For Unauthenticated
- **403** – For Forbidden request
- **404** – For Not Found
- **406** – For Not acceptable
- **425** – For Unsupported Media
- **429** – Too many Requests

File-Uploading in Flask:

File Uploading in Flask is very easy. It needs an HTML form with enctype attribute and URL handler, that fetches file and saves the object to the

desired location. Files are temporary stored on server and then on the desired location.

The HTML Syntax that handle the uploading URL is :

```
form action="http://localhost:5000/uploader" method="POST" enctype="multipart/form-data"
```

d following python code of Flask is:

```
from flask import Flask, render_template, request
from werkzeug import secure_filename
app = Flask(__name__)

@app.route('/upload')
def upload_file():
   return render_template('upload.html')

@app.route('/uploader', methods = ['GET', 'POST'])
def upload_file():
   if request.method == 'POST':
      f = request.files['file']
      f.save(secure_filename(f.filename))
      return 'file uploaded successfully'

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

Sending Form Data to the HTML File of Server:

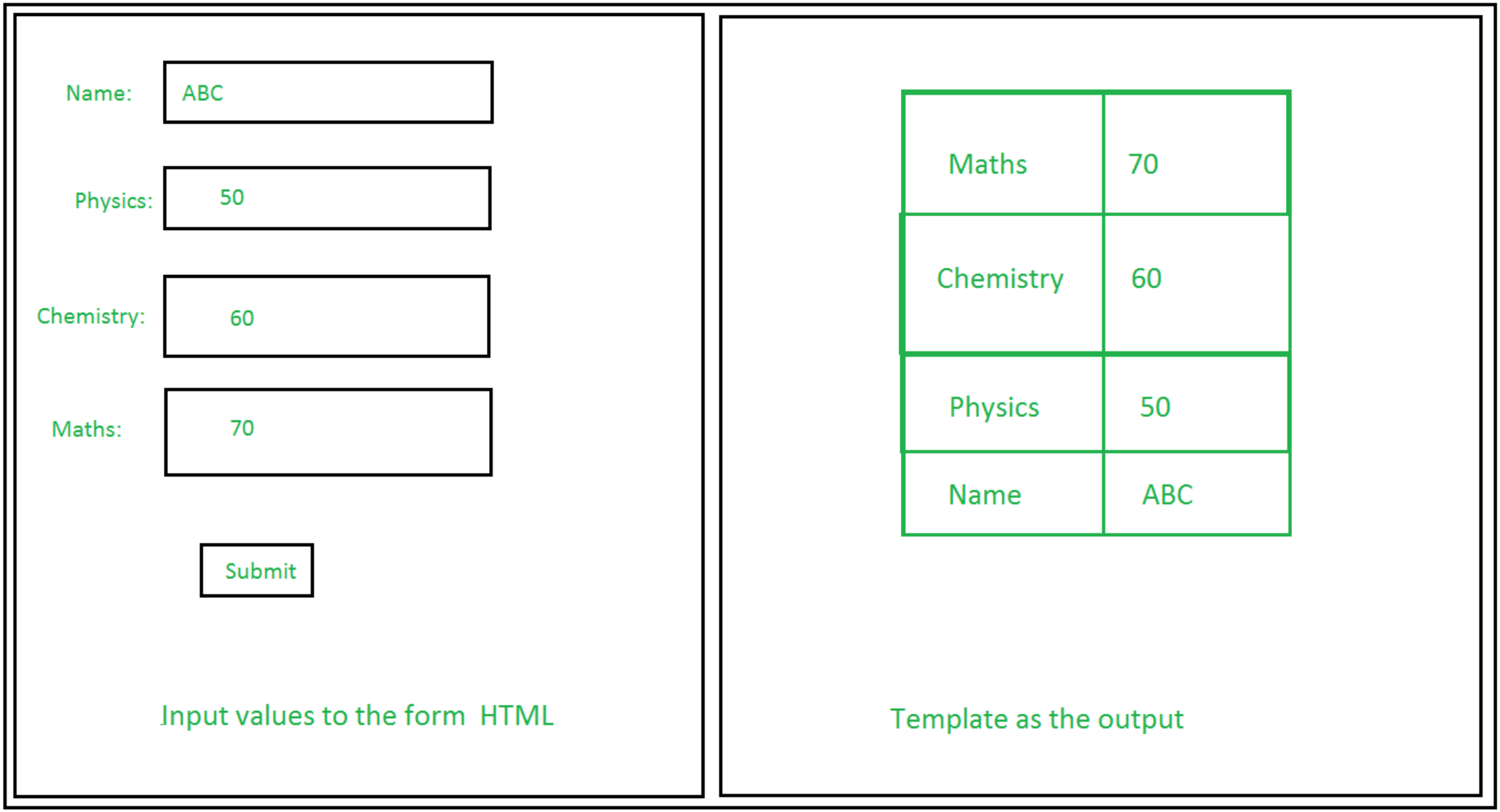
A Form in HTML is used to collect the information of required entries which are then forwarded and stored on the server. These can be requested to read or modify the form. The flask provides this facility by using the URL rule. In the given example below, the '/' URL renders a web page(student.html) which has a form. The data filled in it is posted to the '/result' URL which triggers the result() function. The results() function collects form data present in request.form in a dictionary object and sends it for rendering to result.html.

```
from flask import Flask, render_template, request
app = Flask(__name__)
@app.route('/')
def student():
   return render_template('student.html')

@app.route('/result', methods = ['POST', 'GET'])
def result():
   if request.method == 'POST':
      result = request.form
      return render_template("result.html", result = result)

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

```
<!doctype html>
<html>
   <body>

      <table border = 1>
         {% for key, value in result.items() %}

            <tr>
               <th> {{ key }} </th>
               <td> {{ value }} </td>
            </tr>

         {% endfor %}
      </table>

   </body>
</html>
```

```
<html>
   <body>
```

```
      <form action = "http://localhost:5000/result" method = "POST">
         <p>Name <input type = "text" name = "Name" /></p>
         <p>Physics <input type = "text" name = "Physics" /></p>
         <p>Chemistry <input type = "text" name = "chemistry" /></p>
         <p>Maths <input type ="text" name = "Maths" /></p>
         <p><input type = "submit" value = "submit" /></p>
      </form>
   </body>
</html>
```



Message Flashing:

It can be defined as a pop-up or a dialog box that appears on the web-page or like alert in JavaScript, which are used to inform the user. This in flask can be done by using the method flash() in Flask. It passes the message to the next template.

Syntax: `flash(message, category)`

message is actual text to be displayed and **category** is optional which is to render any error or info.

Example :

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

# /login display login form
@app.route('/login', methods = ['GET', 'POST'])

# login function verify username and password
def login():
   error = None

   if request.method == 'POST':
      if request.form['username'] != 'admin' or \
         request.form['password'] != 'admin':
         error = 'Invalid username or password. Please try again !'
      else:

         # flashes on successful login
         flash('You were successfully logged in')
         return redirect(url_for('index'))
   return render_template('login.html', error = error)
```

Reference: Flask Documentation