assignment=16

July 27, 2023

1 Q1. Explain GET and POST methods.

GET and POST are two of the most commonly used HTTP methods for sending requests from a client (such as a web browser) to a server.

GET method: The GET method is used to request a resource or information from the server. When a client sends a GET request to a server, it is asking the server to return a specific resource, such as a web page or an image, to the client. The GET request can include query parameters in the URL, and these parameters are visible to anyone who can see the request URL. For example, when you type a search query in a search engine, your browser sends a GET request to the server with the search term as a parameter in the URL.

POST method: The POST method is used to submit data to a server to create or update a resource. When a client sends a POST request to a server, it is sending data to the server in the request body. This data can be in the form of key-value pairs or more complex structures, such as JSON or XML. Unlike GET requests, POST requests do not expose the data in the request body in the URL, which can make them more secure for sending sensitive information such as passwords or credit card numbers.

In summary, the main difference between GET and POST requests is that GET requests are used to retrieve data from a server, while POST requests are used to submit data to a server for processing. GET requests include query parameters in the URL, while POST requests include data in the request body.

2 Q2. Why is request used in Flask?

In Flask, the request object is used to handle incoming HTTP requests from clients. The request object is part of the Flask module and provides a way to access data submitted in a form or as query parameters in the URL.

When a client sends a request to a Flask application, the request object is created automatically by Flask and contains all the data associated with the request, such as the HTTP method, headers, form data, and URL parameters. With the request object, you can access this data and use it to generate a response to the client.

Here are some common use cases for the request object in Flask:

Accessing form data: If a client submits a form to a Flask application, the request object can be used to access the data submitted in the form, such as text inputs, checkboxes, and radio buttons.

Accessing URL parameters: If a client includes parameters in the URL, the request object can be used to access these parameters and use them to generate a response.

Handling file uploads: If a client uploads a file to a Flask application, the request object can be used to access the uploaded file and save it to a server.

Authenticating users: The request object can be used to access headers that contain authentication tokens, such as JWTs, which can be used to authenticate users and authorize access to certain parts of the application.

In summary, the request object in Flask provides a way to access and handle incoming HTTP requests from clients, making it a powerful tool for building web applications with Flask.

3 Q3. Why is redirect() used in Flask?

In Flask, the redirect() function is used to redirect the user to a different URL. This function is commonly used to implement the Post/Redirect/Get (PRG) pattern, which is a design pattern used in web development to prevent duplicate form submissions and improve the user experience.

When a user submits a form or performs an action in a web application, the server sends a response back to the client to confirm that the action was successful. However, if the user refreshes the page or clicks the back button, the same action may be performed again, leading to duplicate submissions and possibly unintended consequences.

The PRG pattern helps to prevent this by redirecting the user to a different URL after the action is completed, so that when the user refreshes or navigates back, they are not resubmitting the same data. The redirect() function in Flask is used to implement this pattern by sending a response to the client with a new URL, causing the client to make a new request to the server and load a new page.

Here is an example of how the redirect() function can be used in Flask:

[ ]:
```
from flask import Flask, redirect, url_for, render_template, request

app = Flask(__name__)

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

@app.route('/login', methods=['POST'])
def login():
    # Check user credentials and log them in
    username = request.form['username']
    password = request.form['password']
    # ...
    # Redirect to the user's profile page after login
    return redirect(url_for('profile', username=username))

@app.route('/profile/<username>')
```

```
def profile(username):
    # Show the user's profile page
    return render_template('profile.html', username=username)
```

4 Q4. What are templates in Flask? Why is the render_template() function used?

In Flask, templates are used to generate dynamic HTML pages. A template is a file that contains placeholders for dynamic content, which can be filled in with data at runtime to generate an HTML page that can be sent to the client.

Templates in Flask are usually written in HTML, with additional syntax for placeholders and control structures that allow you to include dynamic content and logic in the page. The most commonly used template engine in Flask is Jinja2, which provides a powerful and flexible way to generate dynamic HTML pages.

The render_template() function is used to render a template and generate an HTML page based on the data provided by the view function. This function takes the name of a template file and any additional data that is needed to generate the page. The template file is usually stored in a templates directory in the Flask project.

5 Q5. Create a simple API. Use Postman to test it. Attach the screenshot of the output in the Jupyter Notebook.

from flask import Flask, jsonify

app = Flask(name)

@app.route('/apiTest', methods=['POST']) def hello(): response = {'message': 'Hello, World!'} return jsonify(response)

if name == 'main': app.run("0.0.0.0")

[ ]:

[ ]: