Restful API & Flask

1. What is a RESTful API?

A RESTful API (Representational State Transfer) is an architecture style for building web services that use standard HTTP methods (GET, POST, PUT, DELETE) and operate over resources represented by URLs.

---

2. Explain the concept of API specification

An API specification defines the rules for interacting with the API—such as endpoints, request/response format, authentication, etc. Examples include **OpenAPI** or **Swagger**, which serve as blueprints for developers.

---

3. What is Flask, and why is it popular for building APIs?

Flask is a lightweight Python web framework. It's popular because:

- Minimal and easy to learn.
- Great for small to medium APIs.
- Extensible with libraries like Flask-SQLAlchemy and Flask-RESTful.

---

4. What is routing in Flask?

Routing maps a URL to a function. For example, visiting `/hello` might call the `hello()` function.

---

5. How do you create a simple Flask application?
python
CopyEdit
```
from flask import Flask
app = Flask(__name__)

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

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

---

6. What are HTTP methods used in RESTful APIs?

- **GET** – Read data
- **POST** – Create data
- **PUT** – Update data
- **DELETE** – Remove data
- **PATCH** – Partially update data

---

7. What is the purpose of the `@app.route()` decorator in Flask?

It maps a URL to a view function.

---

8. What is the difference between GET and POST HTTP methods?

- **GET**: Requests data (parameters in URL).
- **POST**: Sends data to the server (in body), often for creating resources.

---

9. How do you handle errors in Flask APIs?

Use `@app.errorhandler`:

python
CopyEdit
```
@app.errorhandler(404)
def not_found(e):
    return {'error': 'Not Found'}, 404
```

---

10. How do you connect Flask to a SQL database?

Via **Flask-SQLAlchemy**:

python
CopyEdit
```
from flask_sqlalchemy import SQLAlchemy
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///data.db'
db = SQLAlchemy(app)
```

---

11. What is the role of Flask-SQLAlchemy?

An ORM that integrates SQLAlchemy with Flask for easier database operations using Python classes.

---

12. What are Flask Blueprints, and how are they useful?

Blueprints modularize an application. Each feature or module (like auth, admin) can be placed in its own blueprint.

---

13. What is the purpose of Flask's request object?

Access incoming request data like form data, JSON, headers, etc.

python
CopyEdit
```
from flask import request
data = request.get_json()
```

---

14. How do you create a RESTful API endpoint using Flask?
python
CopyEdit
```
@app.route('/api/data', methods=['GET'])
def get_data():
    return jsonify({'key': 'value'})
```

---

15. What is the purpose of Flask's `jsonify()` function?

It converts Python dictionaries into JSON format and sets proper headers.

---

16. Explain Flask's `url_for()` function

Generates a URL from a function name.

python
CopyEdit
```
url_for('hello')  # returns '/'
```

---

17. How does Flask handle static files (CSS, JS, etc.)?

Flask automatically serves files from the `/static` folder:

html
CopyEdit
```
<link rel="stylesheet" href="{{ url_for('static',
filename='style.css') }}">
```

---

18. What is an API specification, and how does it help in building a Flask API?

It provides clear documentation for frontend/backend developers and supports auto-generated clients, tests, and docs (Swagger, Postman).

---

19. What are HTTP status codes, and why are they important in a Flask API?

They indicate response status:

- `200 OK`
- `404 Not Found`
- `500 Internal Server Error`
  They help clients understand what happened with their request.

---

20. How do you handle POST requests in Flask?
python
CopyEdit
```
@app.route('/submit', methods=['POST'])
```

```
def submit():
    data = request.form['name']
    return f"Hello, {data}"
```

---

21. How would you secure a Flask API?

- Use **HTTPS**
- Implement **authentication** (Token, JWT, OAuth)
- Validate input
- Protect against **CSRF/XSS**
- Rate limiting

---

22. What is the significance of the Flask-RESTful extension?

It simplifies building REST APIs by adding class-based views and request parsing.

---

23. What is the role of Flask's session object?

It stores data across requests (e.g., login status). Uses secure cookies.

🔧 Practical Flask Questions

---

1. Create a basic Flask application

Already shown above in Question 5.

---

2. Serve static files

Place files in `/static` folder and reference via `url_for()`.

---

3. Define routes with different HTTP methods

python
CopyEdit
```
@app.route('/data', methods=['GET', 'POST'])
def data():
    if request.method == 'POST':
        return "Posted!"
    return "GET Request"
```

---

4. Render HTML templates in Flask
python
CopyEdit
```
from flask import render_template

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

---

5. Generate URLs using `url_for`
python
CopyEdit
```
url_for('home')  # Outputs '/'
```

---

6. Handle forms in Flask
python
CopyEdit
```
@app.route('/form', methods=['POST'])
def form():
    username = request.form['username']
    return f"Hello, {username}"
```

---

7. Validate form data

Use `WTForms` or basic Python checks:

python
CopyEdit
```
if not request.form['username']:
    return "Username required"
```

---

8. Manage sessions

python

CopyEdit

```
from flask import session
session['user'] = 'John'
```

---

9. Redirect to a route

python

CopyEdit

```
from flask import redirect
return redirect(url_for('home'))
```

---

10. Handle errors like 404

python

CopyEdit

```
@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html'), 404
```

---

11. Structure using Blueprints

python

CopyEdit

```
# in user.py
bp = Blueprint('user', __name__)

@bp.route('/profile')
def profile():
    return "User profile"

# in main app
app.register_blueprint(user.bp)
```

---

12. Define a custom Jinja filter

python
CopyEdit
```
@app.template_filter('reverse')
def reverse_filter(s):
    return s[::-1]
```

---

13. Redirect with query parameters

python
CopyEdit
```
return redirect(url_for('home', msg='welcome'))
```

---

14. Return JSON responses

python
CopyEdit
```
return jsonify({'name': 'John'})
```

---

15. Capture URL parameters

python
CopyEdit
```
@app.route('/user/<username>')
def user(username):
    return f"Hello, {username}"
```

☕ Java + Data Structures

You mentioned "Java + DS" at the end. Please clarify if you want:

- Java equivalents of Flask concepts?
- Java RESTful API with Spring Boot?
- Data Structures (like trees, graphs, etc.) in Java?