Flask Framework and Routing – Detailed Documentation

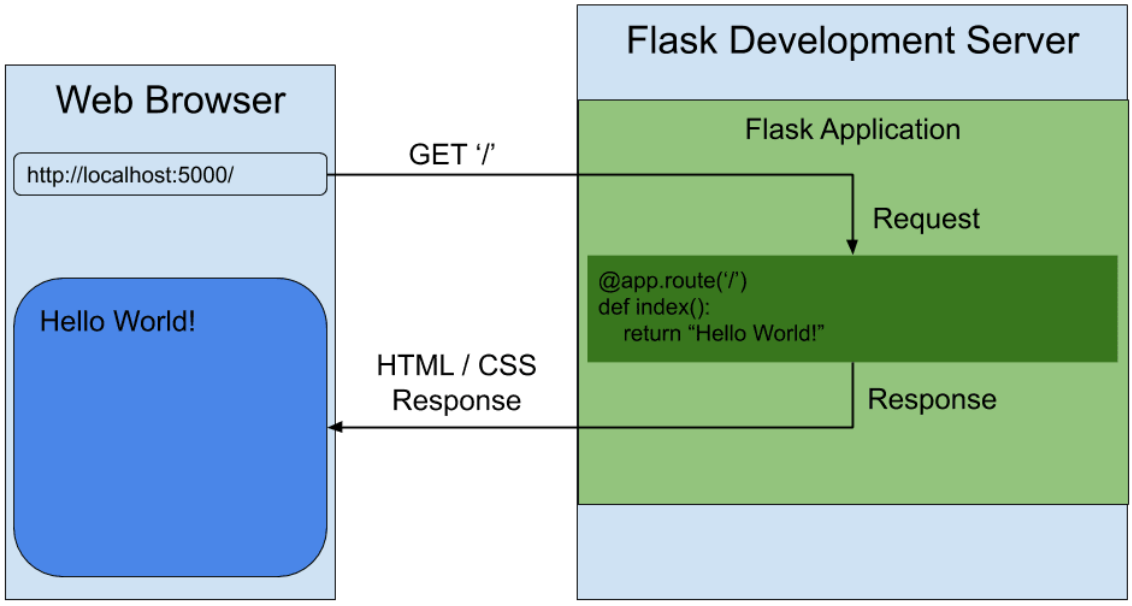
1. Introduction to Flask

Flask is a lightweight, open-source Python web framework used to develop web applications and RESTful APIs. It is classified as a micro-framework because it provides only the core features required for web development, allowing developers to add extensions as needed.

2. Features of Flask

• Lightweight and simple to use
• Built-in development server
• RESTful request handling
• Jinja2 templating engine
• Easy integration with databases
• Supports modular applications using Blueprints

How Flask Works



3. What is Routing in Flask

Routing in Flask refers to the process of mapping URLs to Python functions. Each route is defined using a decorator that binds a URL pattern to a function called a view function.

4. Flask Application Lifecycle

1. Client sends an HTTP request
2. Flask receives the request via WSGI server
3. URL is matched against registered routes
4. Corresponding view function executes
5. Response is returned to the client

5. Basic Route Definition

```
@app.route('/')
def home():
    return 'Welcome to Flask'
```

6. HTTP Methods in Flask Routes

Flask supports multiple HTTP methods:
• GET – Retrieve data
• POST – Create data
• PUT – Update data
• DELETE – Remove data

Methods are defined using the methods parameter in the route decorator.

7. Dynamic Routing

Dynamic routes allow variables in URLs, enabling flexible request handling. These variables can be type-restricted using converters like int, float, or string.

8. Request and Response Objects

Flask provides the request object to access client data such as form values, JSON payloads, query parameters, and headers. Responses can be returned as strings, JSON, or HTML templates.

9. Blueprint and Modular Routing

Blueprints allow developers to organize routes into modules, making large applications easier to manage and maintain.

10. Error Handling in Flask Routes

Flask supports custom error handlers for HTTP errors like 404, 400, and 500. Proper error handling improves application reliability and user experience.

11. REST API Routing Best Practices

• Use RESTful URL naming conventions
• Validate input data
• Return appropriate HTTP status codes
• Handle exceptions gracefully
• Separate concerns using Blueprints

12. Conclusion

Flask routing is a fundamental concept that controls how applications respond to client requests. A strong understanding of routes, HTTP methods, and request handling is essential for building scalable Flask applications and APIs.