Understanding and Using Swagger

Talha Çalışır

3 min read · Just now

Listen Share More

Swagger: Swagger is an open-source framework backed by a large ecosystem of tools that helps developers design, build, document, and consume RESTful web services. It simplifies API development by providing a standard way to describe and interact with APIs.

What is Swagger?

Swagger allows you to describe the structure of your APIs so that machines can read them. The primary purpose of Swagger is to enable API documentation and visualization. The Swagger specification, now known as the OpenAPI Specification (OAS), provides a standard format for documenting APIs. With Swagger, you can:

- Describe endpoints, request parameters, and responses.
- Generate interactive API documentation.
- Facilitate API testing and debugging.

How to Use Swagger?

1. Define Your API: Create a Swagger definition file in YAML or JSON format to describe your API endpoints, methods, request parameters, and response structures.
2. Integrate Swagger UI: Use Swagger UI to visualize and interact with your API's documentation. Swagger UI generates interactive API documentation from the OpenAPI Specification.
3. Add Security Schemes: Define security schemes such as JWT (JSON Web Token) and Basic Auth to secure your API endpoints.

4. Secure Endpoints: Specify which endpoints require authentication using the defined security schemes.

Testing with Swagger

1. Interactive Documentation: Swagger UI allows you to interact with your API directly from the documentation. You can fill in request parameters, send requests, and view responses in real-time.

2. Testing JWT Authentication:

- Obtain a JWT token from your authentication endpoint.
- Use Swagger UI to include the token in the “Authorize” button.
- Test endpoints that require JWT authentication by sending requests with the token.

Testing Basic Auth:

- Use the “Authorize” button in Swagger UI to input your username and password.
- Test endpoints that require Basic Auth by sending requests with the encoded credentials.
- Below is an example of comprehensive API documentation for a simple User Management API using Swagger. This example includes endpoints for managing users, integrating JWT and Basic Authentication.

```yaml
info:
  title: User Management API
  description: API for managing users in the application.
  version: 1.0.0
servers:
  - url: https://api.example.com/v1
    description: Production server

components:
  securitySchemes:
    basicAuth:
      type: http
      scheme: basic
    bearerAuth:
      type: http
      scheme: bearer
```

```
        bearerFormat: JWT
    schemas:
      User:
```

Open in app ↗

Search

```
        name:
          type: string
          example: John Doe
        email:
          type: string
          example: johndoe@example.com
    ErrorResponse:
      type: object
      properties:
        code:
          type: integer
          example: 400
        message:
          type: string
          example: Bad Request

paths:
  /users:
    get:
      summary: Get all users
      security:
        - bearerAuth: []
      responses:
        '200':
          description: A list of users
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/User'
        '401':
          description: Unauthorized
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'
    post:
      summary: Create a new user
      security:
        - basicAuth: []
      requestBody:
        required: true
        content:
          application/json:
```

```
            schema:
              $ref: '#/components/schemas/User'
      responses:
        '201':
          description: User created successfully
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/User'
        '400':
          description: Bad Request
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'
        '401':
          description: Unauthorized
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'

  /users/{userId}:
    get:
      summary: Get user by ID
      security:
        - bearerAuth: []
      parameters:
        - name: userId
          in: path
          required: true
          schema:
            type: integer
      responses:
        '200':
          description: User details
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/User'
        '401':
          description: Unauthorized
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'
        '404':
          description: Not Found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'
```

```
    put:
      summary: Update user by ID
      security:
        - bearerAuth: []
      parameters:
        - name: userId
          in: path
          required: true
          schema:
            type: integer
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/User'
      responses:
        '200':
          description: User updated successfully
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/User'
        '400':
          description: Bad Request
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'
        '401':
          description: Unauthorized
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'
        '404':
          description: Not Found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'

    delete:
      summary: Delete user by ID
      security:
        - bearerAuth: []
      parameters:
        - name: userId
          in: path
          required: true
          schema:
            type: integer
      responses:
```

```
        '204':
          description: No Content
        '401':
          description: Unauthorized
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'
        '404':
          description: Not Found
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'

security:
  - basicAuth: []
  - bearerAuth: []
```

Swagger API Documentation

Edit profile

Written by Talha Çalışır

3 Followers

Software Engineer, Violinist. Androiddo Enthusiast, Action Figure Collector

More from Talha Çalışır