|  |  |
| --- | --- |
| **Topic** | SWAGGER |
| **Document Name** | SWAGGER |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Difficulty Level** | | | |
| **Beginner** | **Junior** | **Senior** | **Expert** |
| □ | ■ | □ | □ |

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Ver | Comments |
| 05.03.2025 | Murat Kağan Temel | 1.0 | Initial Draft |

# SWAGGER

## Exercise SWAGGER:

**Definiton:** Create a layered Spring Boot RESTful application that provides basic CRUD operations for managing Customer data. The application should include OpenAPI-compliant API documentation using the springdoc-openapi-starter-webmvc-ui dependency. This dependency also enables Swagger UI, so no additional configuration is required.

You may choose to simulate database operations using in-memory data structures such as Map or List. Using a real database is not mandatory.

The application should expose endpoints for the following operations:

* **Create a new customer**
* **Retrieve a customer by ID**
* **Retrieve all customers**
* **Update an existing customer**
* **Delete a customer**

After completing the implementation, open the Swagger UI in your browser and share screenshots of the generated API documentation and the available endpoints. A sample CustomerDTO (DTO refers to Data Transfer Object) is shared in the next page.

public class CustomerDTO {

@Schema(description = "Unique ID of the customer", example = "1")

private Long id;

@NotBlank

@Schema(description = "Full name of the customer", example = "John Doe")

private String name;

@Email

@Schema(description = "Email address", example = "john.doe@example.com")

private String email;

}

## SWAGGER:

**Your Answer:**

Öncelikle **Spring Initializr** üzerinden yeni bir Spring Boot projesi oluşturdum. Spring Web, Lombok, Validation, Swagger/OpenAPI destegi için springdoc-openapi-starter-webmvc-ui kütüphanelerini import ettim.

API üzerinden taşınacak müşteri verisini temsil edecek bir CustomerDTO sınıfı oluşturdum. Projede gerçek bir veritabanı kullanmak zorunlu olmadığı için verileri bellekte saklamak amacıyla bir CustomerRepository sınıfı yazdım. Burada bir HashMap kullanarak müşteri listesini tuttum. İş kurallarını ve repository erişimlerini kapsayan CustomerService sınıfını geliştirdim. Bu sınıfta CRUD işlemleri tanımlandı.

Dış dünyaya açılan REST endpoint’lerini CustomerController sınıfında tanımladım.

Bu sınıf aşağıdaki işlemleri sağlar:

* Yeni müşteri oluşturma (POST)
* Müşteri listeleme (GET)
* ID ile müşteri bulma (GET)
* Müşteri güncelleme (PUT)
* Müşteri silme (DELETE)

Swagger dokümantasyonu için springdoc-openapi-starter-webmvc-ui bağımlılığını pom.xml dosyama ekledim. Uygulamayı başlattığımda otomatik olarak Swagger arayüzü aktif hale geldi.

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Yapay zeka tarafından oluşturulmuş içerik yanlış olabilir.