ModelMapper in Spring Boot (Guide + Interview Q&A;)

1. What is ModelMapper?

ModelMapper is a Java library used to map one object to another automatically. In Spring Boot projects, it is mainly used for converting Entity objects into DTOs and DTOs into Entities. This reduces repetitive setter/getter mapping code and keeps the service layer clean.

2. Why do we use DTOs with ModelMapper?

We use DTOs to separate API request/response objects from database entities. This improves security and maintainability. For example, a Request DTO can prevent the user from sending fields like id, createdAt, or internal flags. A Response DTO ensures that only required data is exposed to clients.

3. Small Live Example (Entity → DTO)

Entity:
```
public class Product {
  private Long id;
  private String name;
  private Double price;
}
```

Response DTO:
```
public class ProductResponseDto {
  private Long id;
  private String name;
  private Double price;
}
```

Mapping Code:
```
ProductResponseDto dto = modelMapper.map(product, ProductResponseDto.class);
```

4. Small Live Example (DTO → Entity for Save/Update)

Request DTO:
```
public class ProductRequestDto {
  private String name;
  private Double price;
}
```

Mapping Code:
```
Product product = modelMapper.map(requestDto, Product.class);
```

5. ModelMapper Bean Configuration

To use ModelMapper in Spring Boot, define it as a Bean in a configuration class:

```
@Configuration
public class AppConfig {
  @Bean
  public ModelMapper modelMapper() {
    return new ModelMapper();
  }
}
```

6. Interview Questions and Answers (ModelMapper)

Q: What is ModelMapper in Spring Boot?

A: ModelMapper is a library that helps in mapping objects automatically, mainly used for converting Entity to DTO and DTO to Entity.

Q: Why should we not expose Entity directly in API response?

A: Entities may contain sensitive or unnecessary fields. Using DTO ensures only required data is shared with the client.

Q: What is the difference between Request DTO and Response DTO?

A: Request DTO is used to accept input from client (POST/PUT/PATCH). Response DTO is used to send output to client (GET responses).

Q: How does ModelMapper work?

A: It maps fields automatically based on matching field names and compatible data types.

Q: Can ModelMapper map nested objects?

A: Yes, it can map nested objects, but complex mappings may require custom configuration.

Q: What is the advantage of ModelMapper compared to manual mapping?

A: It reduces boilerplate code, improves readability, and speeds up development.

Q: What are the disadvantages of ModelMapper?

A: It may cause performance overhead in large-scale mapping, and debugging complex mappings can be difficult.

Q: How can we handle different field names between DTO and Entity?

A: We can use custom property mappings or TypeMap configuration in ModelMapper.

Q: Is ModelMapper mandatory in Spring Boot?

A: No, it is optional. Developers can use manual mapping or other libraries like MapStruct.

Q: Which is better: ModelMapper or MapStruct?

A: MapStruct is faster because it generates mapping code at compile time. ModelMapper is easier for quick development but slower at runtime.