How do you map your request data to entities while saving to DB ? What kind of mapper are you using for it?

I map request data (DTOs) to Entities using a dedicated **Mapper Layer**. I **never** expose my JPA Entities directly to the API controller because it creates tight coupling and security risks (like over-posting).

What kind of mapper do I use?

"I use **MapStruct** for object mapping. It is a compile-time code generator, which makes it:

1. **Fast:** It generates plain Java code (no reflection overhead like ModelMapper).
2. **Safe:** If I miss a field mapping, the build fails immediately (compile-time safety).
3. **Clean:** It removes all the boilerplate getter/setter code from my services."

How I implement it (Code Example)

1. The DTO and Entity

```
// UserRequestDTO.java (From API)
public class UserRequestDTO {
    private String firstName;
    private String lastName;
    private String email;
}

// UserEntity.java (To DB)
@Entity
public class UserEntity {
    @Id
    private Long id;
    private String fullName; // Mismatch with DTO
    private String email;
}
```

2. The Mapper Interface (MapStruct)

```
@Mapper(componentModel = "spring") // Makes it a Spring Bean
public interface UserMapper {
    // Custom mapping for field mismatch
    @Mapping(target = "fullName", expression = "java(dto.getFirstName() + ' ' + dto.getLa
    UserEntity toEntity(UserRequestDTO dto);

    UserResponseDTO toResponse(UserEntity entity);
}
```

3. Usage in Service Layer

```
@Service
public class UserService {

    @Autowired
    private UserMapper userMapper; // Inject the generated mapper

    public void createUser(UserRequestDTO request) {
        // 1. Convert DTO -> Entity
        UserEntity entity = userMapper.toEntity(request);

        // 2. Save to DB
        userRepository.save(entity);
    }
}
```

Interview Answer Summary

"I strictly separate my API models (DTOs) from my Database models (Entities). I use **MapStruct** to handle the conversion because it performs better than reflection-based libraries and gives me compile-time safety. I define a `@Mapper` interface, and MapStruct automatically generates the implementation class during the build."

⁂

1. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/429036f4-3bc3-4a7a-8c6f-90246e73b9b9/image.jpg
2. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/24f18c0a-b56e-4862-a619-e2959036a5c2/image.jpg
3. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/97e9ed86-ed11-4035-a18b-1eb05c1bab4e/image.jpg
4. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/87f62423-96f2-4071-9802-8f6699e0ecd8/image.jpg