**Clase DTO, Entidad y capas**

**Entidad**

La **entidad** es una **clase** que representa una **tabla** en la base de datos. La entidad se utiliza para interactuar con la base de datos y realizar operaciones CRUD. Por supuesto, Spring Boot, sus módulos y sus dependencias (Como Spring Data) nos proporcionan anotaciones para el trabajo correcto: @Entity, @Id, @Table

Por ejemplo, si tenemos la entidad Usuario, en la base de datos tendremos una tabla llamada Usuario. Cada atributo de la clase Usuario será un campo de la tabla.

**Clase DTO**

**Clase DTO**

Una clase DTO (*Data Transfer Object*) es un objeto que se utiliza para transferir datos entre diferentes capas de una aplicación. En Spring Boot, las clases DTO se utilizan comúnmente para transferir datos entre la **capa de presentación (controlador, en API Rest)** y la **capa de negocio (service) o** entre la capa de negocio y la **capa de persistencia (repositorio)**.

La clase DTO NO interactúa con la base de datos. De eso se encarga la clase entidad.

**Características**

Las clases DTO sueles tener las siguientes características:

* Atributos. Las clases DTO tienen atributos que corresponden a los datos que se van a transferir.
* Constructores.
* Getter y Setter. Obviamente, la anotación @Data, proporcionada por la librería lombok, me genera esos métodos y más de forma automática

**Ejemplo**

public class UsuarioDTO

{

    @Data //Anotación, proporcionada por la librería lombbok, que me genera de forma automática los métodos getter, setter y más.

    private Long id;

    private String nombre;

    private String email;

    public UsuarioDTO(){} //builder vacío

    public UsuarioDTO(String nombre, String email, int id)

    {

        this.setId(id);

        this.setNombre(nombre);

        this.setEmail(email);

    }

}

**Mapeo**

Para utilizar la clase DTO y la entidad, se necesita realizar un mapeo entre los dos objetos. Esto se puede hacer de dos maneras:  
 - Manualmente. Se puede crear un método que mapee los atributos de la entidad a la clase DTO.

-Utilizando una biblioteca de mapeo. Se puede utilizar una biblioteca como ModelMapper o Dozer para realizar el mapeo entre la entidad y la clase DTO.

**Ejemplo**

A continuación, se presenta un ejemplo de cómo se puede mapear la entidad Usuario a la clase Usuario DTO.

Clase Entidad Usuario

@Entity

@Table("usuarios", schema = 2)

public class Usuario

{

    @Data

    @Id

    private Long id;

    private String nombre;

    private String email;

}

Clase DTO UsuarioDTO

public class UsuarioDTO

{

    @Data //Anotación, proporcionada por la librería lombbok, que me genera de forma automática los métodos getter, setter y más.

    private Long id;

    private String nombre;

    private String email;

    public UsuarioDTO(){} //builder vacío

    public UsuarioDTO(String nombre, String email, int id)

    {

        this.setId(id);

        this.setNombre(nombre);

        this.setEmail(email);

    }

}

Clase de mapeo UsuarioMapper

public class UsuarioMapper

{

    public static  UsuarioDTO toDTO(Usuario usuario)

    {

        UsuarioDTO dto = new UsuarioDTO();

        dto.setId(usuario.getId());

        dto.setNombre(usuario.getNombre);

        dto.setEmail(usuario.getEmail);

        return dto;

    }

}

**Transferencia de datos**

La transferencia de datos entre capas se refiere a la forma en que se pasan los datos de una capa a otra. Por ejemplo:

* Capa de presentación -> capa de negocio. El controlador le puede enviar un objeto UsuarioDTO al servicio para que se realice una operación.

Es que los DTO se utilizan para transferir datos entre capas de la aplicación. Los DTO son objetos que contienen los datos que se necesitan para realizar una operación específica.

* Capa de negocio -> capa de persistencia. El servicio le envía un objeto Usuario al repositorio para que se guarde en la base de datos.

A continuación, se muestra un ejemplo de cómo se puede utilizar un DTO para transferir datos entre la capa de presentación y la capa de negocio:

UsuarioController

@RestController

public class UsuarioController

{

    @AutoWeird //Inyecto dependencias

    private UsuarioService usuarioService;

    @PostMapping("/usuarios")

    public ResponseEntity<UsuarioDTO> crearUsuario(@RequestBody UsuarioDTO usuarioDTO)

    {

        //Llamar a la capa de negocio

        UsuarioDTO usuarioCreado = usuarioService.crearUsuario(usuarioDTO);

        return ResponseEntity.ok(usuarioCreado);

    }

}

UsuarioService