<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.28</version>

<scope>provided</scope>

</dependency>

<https://projectlombok.org/setup/maven>

LOMBOK INSTALAR EN ARCHIVO pom.xml

Anotaciones

//@Setter(value = AccessLevel.PUBLIC)  
  
@Setter  
@Getter  
@AllArgsConstructor  
@NoArgsConstructor  
@FieldDefaults(level = AccessLevel.*PRIVATE*)

@Setter

@Getter  
crea los métodos get y set en la clase, con el modificador (value = AccessLevel.PUBLIC) podemos poner los métodos get o set como privados, públicos, o protectec

@AllArgsConstructor  
constructor con todos los atributos

@NoArgsConstructor

Constructor sin atributos

@FieldDefaults(level = AccessLevel.*PRIVATE*)

Permite poner los atributos privados sin escribir tanto código y modifica el acceso de aquellos atributos que no tengan declarado el modificador de acceso

También tiene la posibilidad de poner final a los atributos

@FieldDefaults(makeFinal = true)

@ToString

Para poder devolver los datos de un objeto como un string

@ToString(exclude = {"apellido","edad"})

Permite excluir los atributos que nos parezca necesario

@ToString(includeFieldNames = false)

Permite imprimir como string sin los nombres de filas

Ejemplo Persona(id=1, nombre= Carlos) con el atributo seria:

Persona(1,Carlos)

@ToString(of = {"nombre","apellido"})

Permite imprimir solo los datos que se incluyen en el metodo of por ejemplo Persona(Carlos,Rodriguez)

@ToString(of = {"nombre","apellido"},includeFieldNames = false)

Se puede combinar

@ToString(callSuper = true)

En caso de herencia regresa los datos de la clase padre…

@EqualsAndHashCode

Me permite crear los metodos Equal y HashCode

Atributos de @EqualsAndHashCode

@EqualsAndHashCode(exclude = {"nombre"})

Puedo excluir algún atributo de la creación de los metodos

@EqualsAndHashCode(of = {"nombre","apellido"})

Crear los metodos solo de los atributos nombrados

@EqualsAndHashCode(doNotUseGetters = true)

No usara los getter para generar el método pero si los atributos. Por defecto es flase.

@Data

Crea get y set, constructor vacio, Equal, HashCode, toString

Estructura comun en proyectos

@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@FieldDefaults(level = AccessLevel.*PRIVATE*)  
public class Persona {  
  
 Long id;  
 String nombre;  
 String apellido;  
 String email;  
 Integer edad;  
 Integer telefono;  
 LocalDate fechaDeNacimiento;  
  
}

Notacion @Builder

Patrón de diseño de objetos builder INVESTIGAR

@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@FieldDefaults(level = AccessLevel.*PRIVATE*)  
@Builder  
public class Persona {  
  
 Long id;  
 String nombre;  
 String apellido;  
 String email;  
 Integer edad;  
 Integer telefono;  
 LocalDate fechaDeNacimiento;  
  
}

Se usa para construir un objeto persona por ejemplo con las propiedades que quisiéramos tener de la siguiente forma

Persona persona1= Persona  
 .*builder*()  
 .id(2L)  
 .nombre("Juan")  
 .apellido("Ramirez")  
 .edad(23)  
 .build();

**Atributos para builder**

@Builder(access = AccessLevel.*PUBLIC*)

**Configurar acceso**

@Builder(builderMethodName = "create")

**Cambia el nombre de metodo para construir el objeto. Builder().**

@Builder(buildMethodName = "construir")

**Cambia el nombre del método buil().**

@Builder(builderClassName = "personConstructor")

**Cambiar el nombre de la clase**

**Notación para DTO para clases inmutables**

@Value

**Crea getter constructor con todos los atributos, equals, hashCode, toString, atributos privados y finales**

@RequiredArgsConstructor

**Obliga a agregarle valores a los atributos marcados como privados y final**

**El requiredArgsConstructor con el Data y NotNull crea un constructor con los atributos marcados con el NotNull y también a los atributos marcados como final**

@NonNull

**No permite el atributo nulo**

@Log

**Permite imprimir en distintas partes del programa**

*log*.info(persona.toString());

*log*.warning(persona.toString());

@Slf4j

**Otra forma más recomendada de Log**

**Se debe importar desde el pom.xml**

<dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-classic</artifactId>  
 <version>1.4.7</version>  
</dependency>

@Cleanup

**Permite cerrar recursos que extienden de la class Closeable**

**Liberar espacio en memoria**

@SneakyThrows  
public static void main( String[] args ) {

**Permite con una sola línea de código manejar throws propaga la excepción**