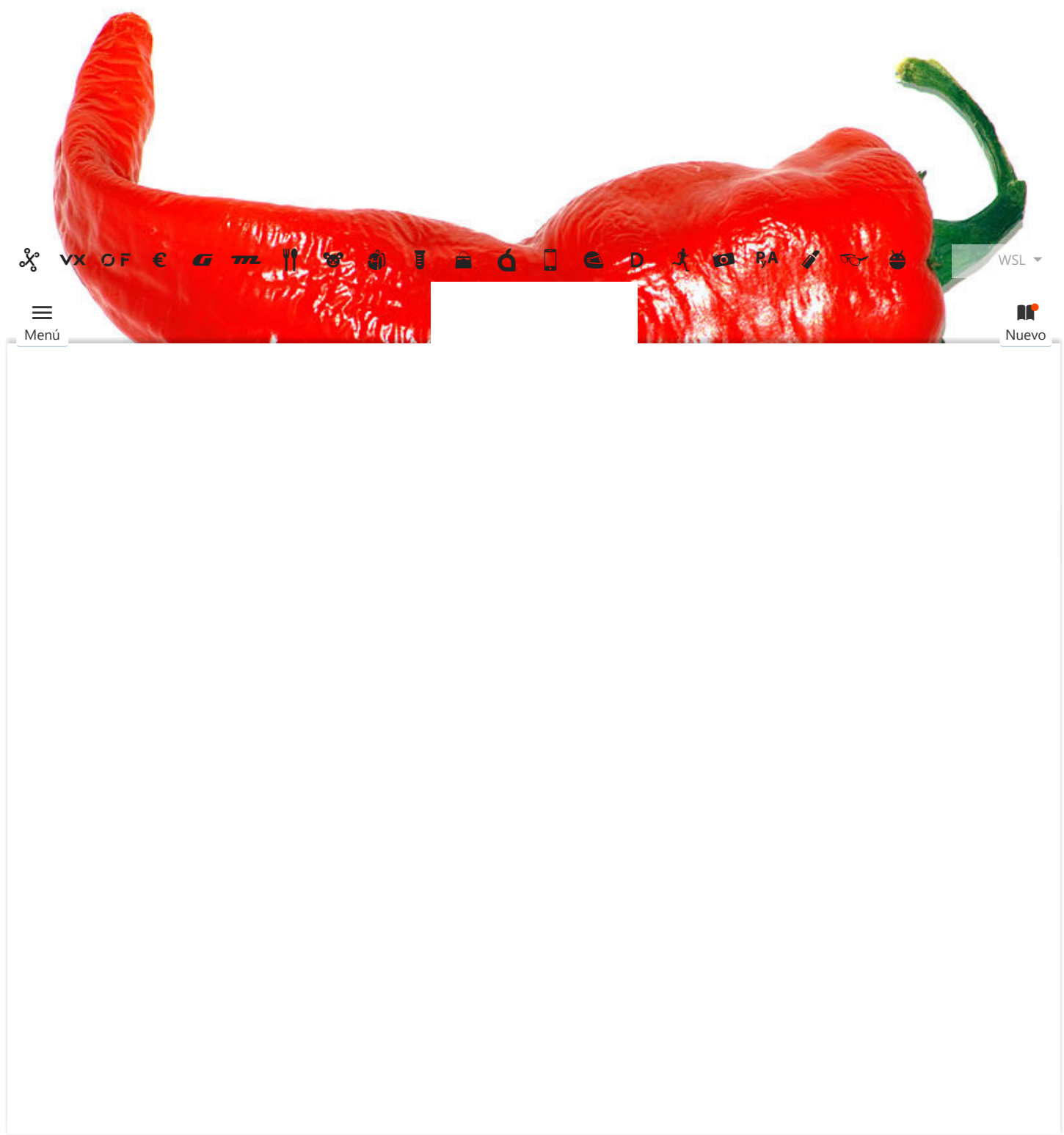


Simplificando Java con Lombok

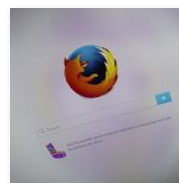


Compartir

"Simplificando Java con Lombok"



TE RECOMENDAMOS



Firefox 52 te deja enviar pestañas del PC al móvil y dice adiós a Java, pero no a Flash



Principios de una arquitectura limpia: mantenible y testeable



Cómo usar KotlinJS como sustituto de Javascript para desarrollo Web

PUBLICIDAD

Desliza para ver más »

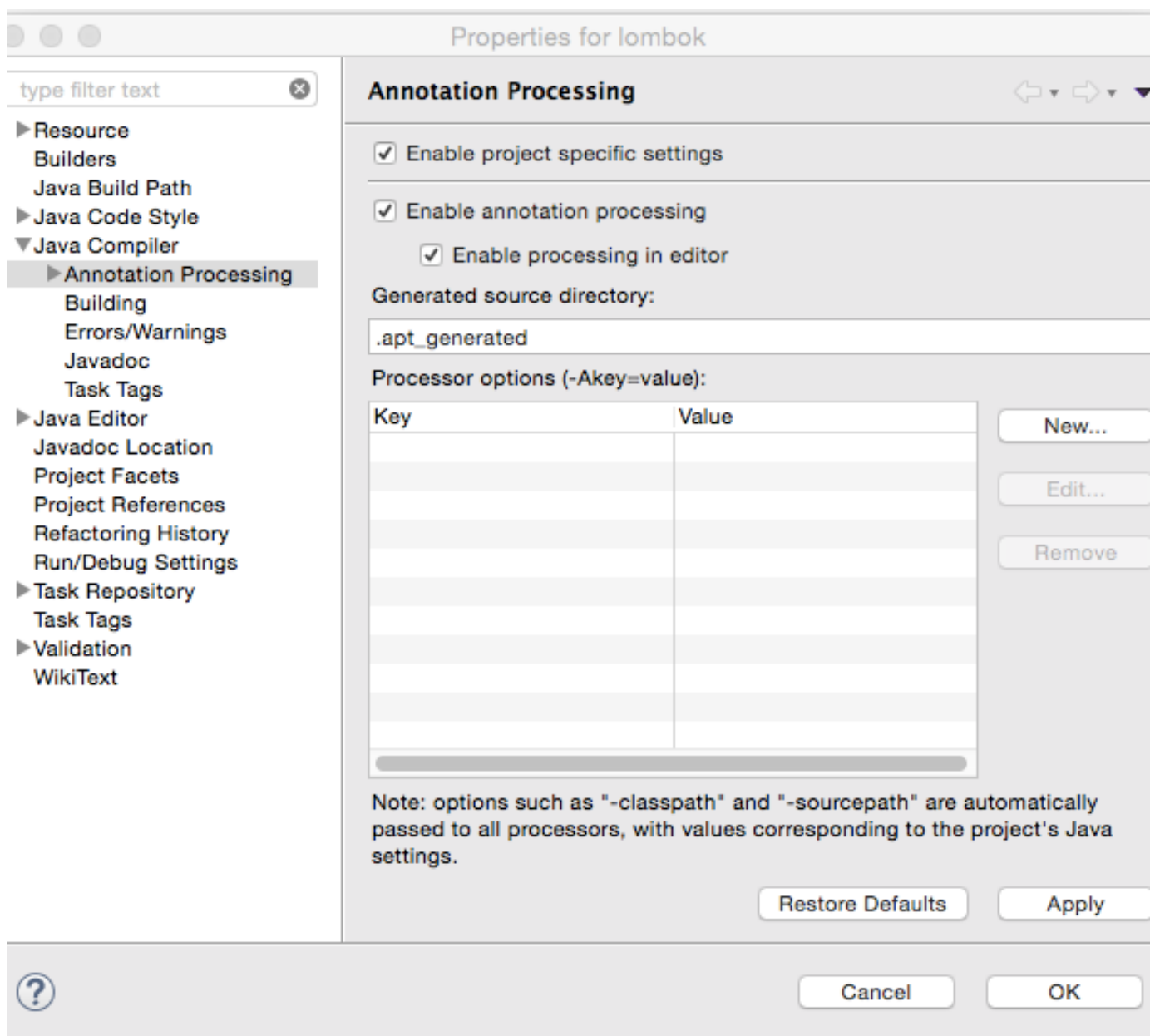
```
1 package es.genbetadev1;
2
3 public class Persona {
4
5     private String nombre;
6     private String apellidos;
7     private int edad;
8 }
```

Esta clase no es operativa y hay que usar las diferentes herramientas de generación de código para convertirla en esto :

Desliza para ver más »

```
1 package es.genbetadev2;
2
3 public class Persona {
4
5     private String nombre;
6     private String apellidos;
7     private int edad;
8     public String getNombre() {
9         return nombre;
10    }
11    public void setNombre(String nombre) {
12        this.nombre = nombre;
13    }
14    public String getApellidos() {
15        return apellidos;
16    }
17    public void setApellidos(String apellidos) {
18        this.apellidos = apellidos;
19    }
20    public int getEdad() {
21        return edad;
22    }
23    public void setEdad(int edad) {
24        this.edad = edad;
25    }
26    public Persona(String nombre, String apellidos, int edad) {
27        super();
28        this.nombre = nombre;
29        this.apellidos = apellidos;
30        this.edad = edad;
31    }
32
33 }
```

hay que configurar el entorno de desarrollo para que pre-processe las anotaciones. En el ejemplo se muestra una de las siguientes menús:



Finalizado este paso podemos definir la clase Persona apoyándonos en las anotaciones de Lombok.

Desliza para ver más »

```

1 package es.genbetadev3;
2
3 import lombok.Data;
4
5 @Data class Persona {
6
7     private String nombre;
8     private String apellidos;
9     private int edad;
10
11 }
12

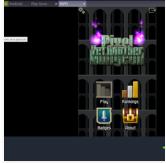
```

Ya no será necesario añadir para nada los métodos set/get o los constructores, lombok los añadirá de forma dinámica y la clase Persona funcionará sin problemas.

PUBLICIDAD

TAMBIÉN TE PUEDE GUSTAR

Creación de aplicaciones React "The Agile Way"



Cómo utilizar aplicaciones de Android en Windows



Ethereum, así es la "promesa" de las criptomonedas que quiere convertirse en el nuevo Bitcoin



Las 16 mejores series de estreno de 2016



Smartwatch Asus ZenWatch 2 con 30 euros de descuento y envío gratis



Por qué un editor de texto de hace 40 años machaca al "todopoderoso" Atom



Firefox 52 te deja enviar pestañas del PC al móvil y dice adiós a Java, pero no a Flash

! Comentarios cerrados

22 comentarios

OPCIONES



webstilos

28 Jun. 2015, 13:00 #12

Yo trabajo con lombok y es un lujo. De pasar a tener una clase llena de código (que obviamente es necesario) y solo va a hacer que crecer tus ficheros, a con una simple anotación generar todo esto en tiempo de compilación, es una genialidad. Ya no solo getters y setters, sino de 'atajos' como el val de lombok, a cerrar streams con @Cleanup o a usar patrones como @Delegate, mejoran la legibilidad a la vez que la productividad.



miguelangellv

27 Jun. 2015, 8:53 #5

Los gets y sets permiten añadir control al acceso a las variables. Por ejemplo evitar que una distancia sea negativa o marcar que valores han sido modificados y hay que actualizar la BD.

También permite hacer transparente los datos guardados o los calculados, como una edad o un área.



Respondiendo a
miguelangellv

CERRAR RESPUESTAS ×



antoniodomene

27 Jun. 2015, 19:22 #9

Estoy totalmente de acuerdo contigo. El acceso directo a las variables miembro es claramente un defecto de diseño o mal uso del lenguaje y atenta contra los principios de diseño de JAVA. A largo plazo puede provocar errores y dificultar la lectura del código resultante.



Respondiendo a
miguelangellv

forumisto

**Carlitos Way**

27 Jun. 2015, 13:00 #6

Bueno, desde hace muchos años Eclipse tiene la opción de crear Getters y Setters de manera automática según los campos, y sin librerías, jars ni dependencias de por medio.

La verdad es que del mundo Java a veces me sorprende cómo se complican la vida y sobre todo la variedad de metodologías con las que se programa, bastante desconcertante si lo comparas con RoR, Symfony, o incluso .NET.

**s_48k**

27 Jun. 2015, 14:50 #8

Hombre, usen groovy. Si los get y los set no se van a modificar no hace falta ponerlos. Si se hace algún tipo de gestión dentro de ellos, entonces hay que ponerlos.

**Gerardo**

28 Jun. 2015, 3:46 #11

Esto empieza a sonar a secta

**fernandoenzo**

27 Jun. 2015, 13:26 #7

Para los que me habéis respondido: Habitualmente programo en Python y últimamente en PHP bastante.

Y repito lo que dije antes: No estoy desterrando completamente los getters y setters. Únicamente los que son manifiestamente inútiles frente a un atributo público, como los del ejemplo expuesto en el post. Soy perfectamente consciente y he hecho uso a menudo de restricciones en las variables a través de los setters, como por ejemplo que una edad deba ser positiva y cosas así.

Un saludo!

**tukosito**

30 Jun. 2015, 2:08 #14

Joder como esta el patio. De una docena de comentarios solo dos o tres tienen sentido el resto son:

No, si al final lo de las carnicas unos cuantos se lo tienen merecido...



tukosito

30 Jun. 2015, 2:12 #15

Lombok no es un framework, solo una libreria. Y si, por supuesto que se deben utilizar la menor cantidad posible de "acessors", pero eso no quita que los que necesites aun hay que generarlos.

PD: Los getter y setter no se utilizan nunca en el constructor salvo que sean declarados "final".



Respondiendo a
tukosito

CERRAR RESPUESTAS ×



naveto

2 Jul. 2015, 10:28 #21

En el constructor hay que usar los setter para así poder validar los datos que se introducen.

Si declaras un método como "final" lo que le estás indicando es que es un método de clase y que puedes usar sin necesidad de crear un objeto.

Solo es obligatorio usar métodos "final" si son llamados desde otro método "final" y ambos están declarados en la misma clase. Es lo que ocurre si en la clase donde está el main llamas a algún método auxiliar creado en dicha clase, que obligatoriamente tiene que ser "final".

Es más, si lo intentas el compilador va a protestar al declarar esos métodos.



Respondiendo a
naveto

CERRAR RESPUESTAS ×



logui

3 Jul. 2015, 11:16 #22

[quote]

Si declaras un método como "final" lo que le estás indicando es que es un método de clase y que puedes usar sin necesidad de crear un objeto.

[/quote]

final en un método significa que no puede ser sobrescrito en clases hijas.

Y un método de clase se declara con static.



logui

30 Jun. 2015, 14:34 #18


```
maria.deber()
```

en lugar de:

```
jose.getPiernas()  
maria.getBoca()
```

y luego usar esas propiedades internas para nuestra lógica situada en un sitio diferente.

Porque el tema es que entonces jose y maria no serian objetos(encapsulación de datos y lógica) si no estructuras de datos (modelo anémico) y todo pasaría a tener un tufillo procedimental en lugar de orientado a objetos.

Otro tema es que necesites getters y setters para integrar tus clases con los diferentes frameworks que pululan por el ecosistema java, por eso de la especificación de javabeans. O que necesites una estructura de datos en lugar de un objeto, para transferir datos etc..

**fernandoenzo**

26 Jun. 2015, 23:32 #1

Mira que llevo años programando, y siempre me causa gran desconcierto esta, en mi opinión, estupidez de patrón.

¿Para qué narices creo un getter y un setter de un objeto si puedo ponerlo directamente público? No tiene absolutamente ningún sentido salvo el de rellenar con código inútil el archivo java de turno.

Otro tema es que quiera ponerlo únicamente visible pero no modificable (como las muy útiles unmodifiable collections) o alguna variante similar. Pero hacer getters y setters para variables que son Integer, String, Boolean... sencillamente me parece absurdo.

Respondiendo a
fernandoenzo

CERRAR RESPUESTAS ×

**jbono94**

27 Jun. 2015, 6:45 #3

En que lenguaje o paradigma programas habitualmente? Los accessors methods vienen de la programación orientada a objetos, donde los objetos encapsulan su estado interno pero proveen de una interfaz (los accessors) para poder modificar/conocer sus atributos. Y todo sucede mediante un envío de mensajes, no simples asignaciones como en lenguajes como C. Además los setters y getters te dejan incluir lógica dentro de ellos ya que son métodos.

Respondiendo a
jbono94

CERRAR RESPUESTAS ×

**matiasliwski**

27 Jun. 2015, 7:19 #4

Es mi primer respuesta en mi historia en genbetadev, así que tenganme paciencia ;D ... Primero que nada felicitaciones a los creadores y gracias por compartir el conocimiento, ahora la parte que no les va a gustar :P ... en mi opinión antes de

solo se utilizan en la construccion) ... me suena a que que estamos ensuciando el contrato.

Claramente lo que digo no aplica a todos los casos, pero noto que cuando codeo me pasa muchas veces que simplemente agrego los accesors porque "los voy a utilizar" y termine cumpliendo mi propia profecía, redundando en lo que entiendo es un mal diseño...

Entiendo esto, mas allá del lenguaje utilizado para expresar el paradigma...
Espero que aporte y perdón si moleste a alguien con mi comentario.



Respondiendo a
jbono94



forumisto

30 Jun. 2015, 15:30 #20

que sí, si la teoría todo el mundo la entiende.

pero la práctica te dice que son para escribir o leer un valor. si el getter o el setter tiene más intringulis, pues adelante, se implementa y punto, pero entre un `persona.nombre="pepe"` y un `persona.setNombre("pepe")` poca diferencia hay (a no ser que en el método `setNombre` se haga algo más).



Respondiendo a
fernandoenzo



thebronx

29 Jun. 2015, 10:57 #13

opino igual, una clase con getters y setters "generados" hace exactamente lo mismo que una clase (o estructura más bien) con atributos públicos.

los getters/setters tienen sentido cuando hacen cosas, cuando tienen lógica. Peeero, si un getter tiene lógica, ya no es un getter, ya es otra cosa. Y el setter lo mismo, ya no setea una variable, lo mismo modifica una existente, o varias, o modifica el comportamiento de otros métodos... eso ya no es un setter.

Aun así, puestos a hacer getters/setters inútiles, si al menos los hace una librería pues oye, es como generarlos automáticamente solo que sin verlos, tampoco hay mucha diferencia.



Respondiendo a
fernandoenzo



seitoku

27 Jun. 2015, 2:25 #2

Bueno, yo tambien era de los que pensaba asi, pero actualmente trabajo para un proyecto de un banco en el que estan dos equipos diferentes: los del banco en sí que se encargan de la parte de seguridad (entre otras cosas) y nosotros que desarrollamos las funcionalidades en sí.

Entonces, ellos desarrollaron una clase `Cliente` que pasa por su modulo de seguridad (que nosotros no tenemos acceso) y

La cosa es que despues ellos se dieron cuenta que la clase Cliente iba a necesitar eso, saber que producto tenia activo para mostrar el modulo de contraseña y agregaron sus metodos (getProductoActivo, setProductoActivo). En ese momento por las fechas no podiamos cambiar todas las llamadas que ya haciamos asi que simplemente cambiamos la logica interna del getTarjetaActiva y setTarjetaActiva para que apuntara a los metodos que ellos habian creado.

Por cierto, getTarjetaActiva retornaba la clase Tarjeta que extendia de la interfaz Producto (que se lo inventaron tambien despues de que ya habiamos programado varias funcionalidades).



Respondiendo a
fernandoenzo



tukosito

30 Jun. 2015, 2:19 #16

Supongo que llevas programando C o ensamblador desde entonces, no?

Sinceramente, no hay un solo libro de principiante en programacion orientada a objetos que no te explique el por que de "esa estupidez de patron".



franciscocastillo

30 Jun. 2015, 9:43 #17

A ver, no nos vayamos a los extremos.

La mayoría de los getters y setters solo son encapsulaciones que devuelven el valor de una variable privada de la clase, para ese 95% de los casos una librería como Lombok es genial, porque reduce considerablemente la cantidad de código (y lo hace más legible). Que sí, que los getters y setters se generan solos a través del IDE y que no cuesta nada, pero es más fácil trabajar con una clase de 100 lineas que con una de 200 con getters y setters (por no hablar de los "amigos" que te ponen los getters y setters desordenados, o dispersos, o al principio de la clase!)

Y para los casos en los que necesites hacer algo más que devolver una variable (comprobar que la edad no sea negativa, o lo que sea), pues lo sobre-escribes, o en esa clase no usas Lombok y lo haces como hasta ahora... ¿dónde está el problema?



plantasyremedios

28 Jun. 2015, 3:21 #10

Interesante





RECOMENDADO EN MAGNET



Puentes inutilizados y viejos pueblos que brotan del agua: la España de la sequía en 16 fotografías



Todo lo que se ha sabido sobre la Gürtel y el PP mientras todos mirábamos a Cataluña

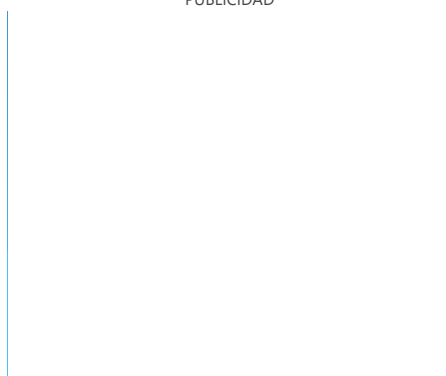


La rara historia tras las banderas de Liberia, las únicas que podrías haber hecho tú mismo en Paint



El motivo psicológico por el que los puntos finales te cabrean tanto en WhatsApp

PUBLICIDAD



RECIBE UN EMAIL AL DÍA CON LOS ARTÍCULOS DE GENBETA DEV:

Tu correo electrónico

SUSCRIBIR

Síguenos



EN GENBETA DEV HABLAMOS DE...

- Curiosidades

Frameworks


Eventos para Desarrolladores
- Open Source

Herramientas de desarrollo

java

VER MÁS TEMAS

Buscar en Genbeta Dev



SUBIR ▲

TECNOLOGÍA	ESTILO DE VIDA	MOTOR	ECONOMÍA	OCIO
Xataka	Tendencias	Motorpasión	El Blog Salmón	Espinof
Xataka Móvil	Tendencias Belleza	Motorpasión Moto	Pymes y Autónomos	Diario del Viajero
Xataka Foto	Tendencias Hombre			
Xataka Android	Directo al Paladar			
Xataka Smart Home	Bebés y Más			
Xataka Windows	Vitónica			
Xataka Ciencia	Decoesfera			
Applesfera				
Vida Extra				
Genbeta				
Genbeta Dev				
Magnet				
Compradicción				
Xataka eSports				

LATINOAMÉRICA				
Xataka México	Directo Al Paladar México	Motorpasión México		
Xataka Colombia				

PARTICIPAMOS EN				
Nobbot	Mi Mundo Philips	Circula Seguro	En Naranja	Bluemagazine
Tecnología de tú a tú	Muy Saludable de Sanitas	Circula Seguro PT	Sage Experience	
Blog Lenovo	Coca-Cola Journey España	Corriente Eléctrica	Bloggin Zenith	
eSports Unlocked by Orange	Coca-Cola Journey México		Seguros de tú a tú	
Inget by acer	Coca-Cola Journey Portugal			
	Zona Coca-Cola			
	Cervezas Alhambra			
	Mahou Rentabilibar			

