Tabla de Contenidos

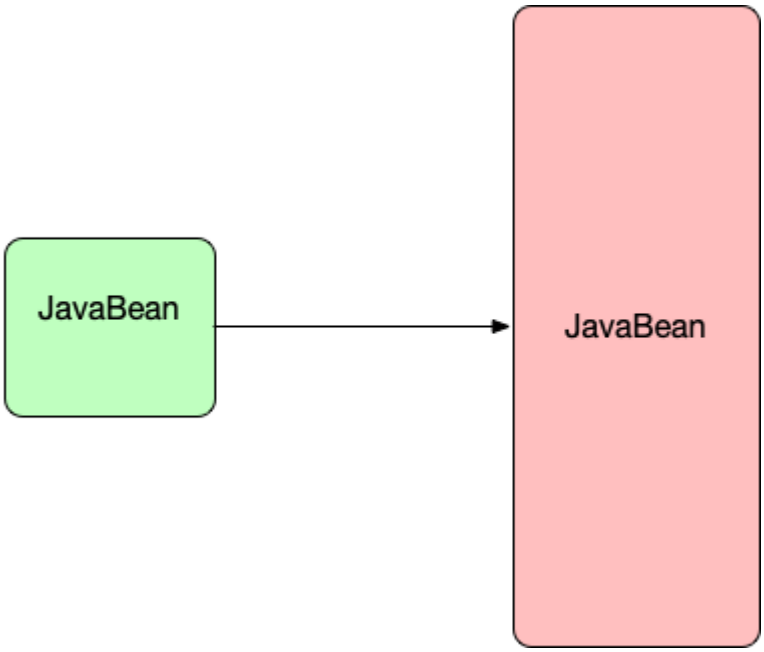
Java @Data es una de las anotaciones más habituales que nos podemos encontrar en proyectos Java que tienen JavaBeans con un monton de propiedades. Normalmente en estos casos es muy habitual usar el proyecto Lombok para reducir la construcción de código que nosotros necesitamos hacer ya que sino se hace infinita . Un ejemplo posible sería una clase como la siguiente:

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Persona {
    @Id
    private String dni;
    private String nombre;
    private int edad;

    private final List<Libro> libros= new ArrayList<>();

    public Persona(String dni) {
        this.dni = dni;
    }
}
```

Java @Data y simplificación

Esta clase tiene solamente las variables privadas y un par de constructores . Eso sí dispone de Java @Data una anotación del proyecto Lombok que preprocesa nuestros JavaBeans y les

añade métodos adicionales .



En este caso es cuestión de ver la documentación o mejor usar un decompilador online de Java para poder ver como esta clase que es tan pequeña y a la cual le he eliminado los imports y la declaración de packages se convierte en tiempo de compilación en una clase mucho más completa que contiene todos los métodos set/get al anotarse con @Data.

```
package es.avalon.dominio;

import java.util.ArrayList;
import javax.persistence.OneToMany;
import java.util.List;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Entity;
```

```
public class Persona
{
    private String dni;
    private String nombre;
    private int edad;
    private final List<Libro> libros;
    public Persona(final String dni) {
        this.libros = new ArrayList<Libro>();
        this.dni = dni;
    }
    public String getDni() {
        return this.dni;
    }
    public String getNombre() {
        return this.nombre;
    }
    public int getEdad() {
        return this.edad;
    }
    public List<Libro> getLibros() {
        return this.libros;
    }
    public void setDni(final String dni) {
        this.dni = dni;
    }
    public void setNombre(final String nombre) {
        this.nombre = nombre;
    }
    public void setEdad(final int edad) {
        this.edad = edad;
    }
```

```
@Override
public boolean equals(final Object o) {
    if (o == this) {
        return true;
    }
    if (!(o instanceof Persona)) {
        return false;
    }
    final Persona other = (Persona)o;
    if (!other.canEqual(this)) {
        return false;
    }
    if (this.getEdad() != other.getEdad()) {
        return false;
    }
    final Object this$dni = this.getDni();
    final Object other$dni = other.getDni();
    Label_0078: {
        if (this$dni == null) {
            if (other$dni == null) {
                break Label_0078;
            }
        }
        else if (this$dni.equals(other$dni)) {
            break Label_0078;
        }
        return false;
    }
    final Object this$nombre = this.getNombre();
    final Object other$nombre = other.getNombre();
    Label_0115: {
```

```
        if (this$nombre == null) {
            if (other$nombre == null) {
                break Label_0115;
            }
        }
        else if (this$nombre.equals(other$nombre)) {
            break Label_0115;
        }
        return false;
    }
    final Object this$libros = this.getLibros();
    final Object other$libros = other.getLibros();
    if (this$libros == null) {
        if (other$libros == null) {
            return true;
        }
    }
    else if (this$libros.equals(other$libros)) {
        return true;
    }
    return false;
}
protected boolean canEqual(final Object other) {
    return other instanceof Persona;
}
@Override
public int hashCode() {
    final int PRIME = 59;
    int result = 1;
    result = result * 59 + this.getEdad();
    final Object $dni = this.getDni();
```

```
        result = result * 59 + (($dni == null) ? 43 :
$dni.hashCode());
        final Object $nombre = this.getNombre();
        result = result * 59 + (($nombre == null) ? 43 :
$nombre.hashCode());
        final Object $libros = this.getLibros();
        result = result * 59 + (($libros == null) ? 43 :
$libros.hashCode());
        return result;
    }
    @Override
    public String toString() {
        return "Persona(dni=" + this.getDni() + ", nombre=" +
this.getNombre() + ", edad=" + this.getEdad() + ", libros=" +
this.getLibros() + ")";
    }
    public Persona() {
        this.libros = new ArrayList<Libro>();
    }
    public Persona(final String dni, final String nombre, final int
edad) {
        this.libros = new ArrayList<Libro>();
        this.dni = dni;
        this.nombre = nombre;
        this.edad = edad;
    }
}
```

JavaBean y su nuevo código

Como se puede observar la clase pasa a ser gigante pero a nosotros nos es totalmente transparente nos basta con modificar el POM.xml y añadir una linea de preprocesado y

Lombok hará el resto.

```
<plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.0</version>
          <configuration>
            <annotationProcessorPaths>
              <path>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <version>1.18.24</version>
              </path>
            </annotationProcessorPaths>
          </configuration>
        </plugin>
```

La anotación Java @Data nos ha generado de forma totalmente transparente los métodos set/get , los métodos Equals y HashCode apoyandose en todos las propiedades que no son finales . Ha añadido el siempre útil método de toString() . Realizando algunas pequeñas optimizaciones de código como cambiar el momento en el que el ArrayList se construye. Todo esto lo hace de forma totalmente transparente para el programador . Eso sí siempre hay que tener en cuenta como funciona la librería y de que forma puede ayudarnos.

Otros artículos relacionados

1. Java Lombok , clases y productividad
2. Java Record Class y JDK 14
3. El concepto de Java Annotations y su funcionamiento