INSTRUCTOR: Goicoechea Forte, Lucas

EJERCICIO GUIADO, MAVEN-LOMBOK-TEST

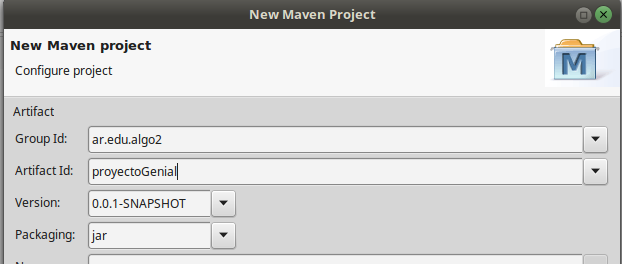
1. Creando proyecto Maven sin arquetipo

Desde el menú principal del Eclipse seleccionamos File > New… Project… Maven Project. En la primera pantalla del asistente

* podemos elegir un [archetype](https://maven.apache.org/guides/introduction/introduction-to-archetypes.html), o bien chequear la opción “Create a simple project (skip archetype selection)” que es la opción por defecto que vamos a elegir.
* Nombre del Proyecto: ProyectoUsuario-LGoicoechea <- aquí su nombre

Al presionar Next, nos aparece el siguiente paso, donde debemos elegir

* el **Group Id** refleja la organización para la que vamos a construir el proyecto (por lo general depende de la materia que estás cursando)
* el **Artifact Id** que se asocia al nombre del proyecto
* la versión, donde dejamos el valor por defecto



Paso 2: Agregar bibliotecas necesarias al pom

En el archivo pom.xml del raíz del proyecto podemos agregar bibliotecas a nuestro proyecto en el nodo dependencies…

<dependencies>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.13</version>

</dependency>

<dependency>

<groupId>commons-collections</groupId>

<artifactId>commons-collections</artifactId>

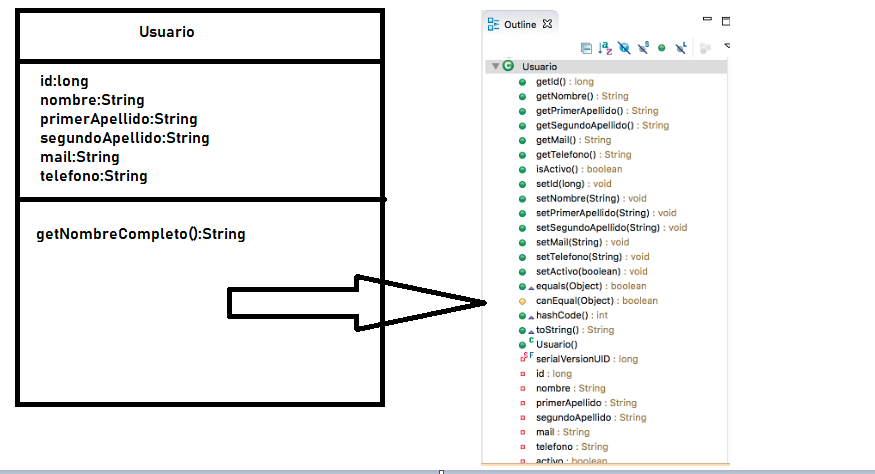
<version>3.1</version>

</dependency>

</dependencies>

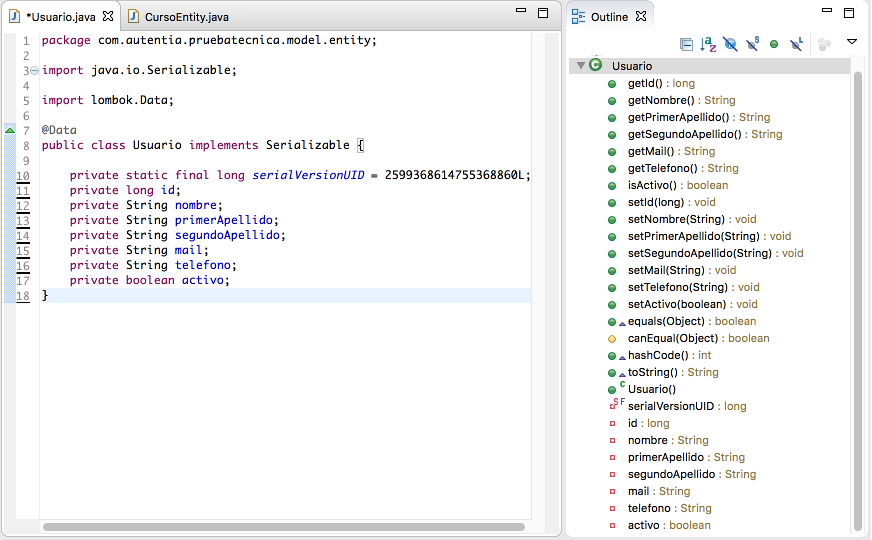
También podemos cambiar el groupId, artifactId, la versión o bien apuntar a un [parent project](https://maven.apache.org/guides/introduction/introduction-to-the-pom.html" \l "Project_Inheritance_vs_Project_Aggregation) (para mayor información consulte con el docente de su materia)

2. Creando clase entidad Usuario



3. Un primer acercamiento

Primero veamos qué puede hacer Lombok por nosotros y luego veamos cómo conseguirlo.

[](http://www.adictosaltrabajo.com/wp-content/uploads/2016/01/lombok_01.png)

En la imagen anterior se observa como apenas hemos declarado la clase y unas pocas propiedades. Sin embargo, en la ventana de *outline* de Eclipse se ven todos los *getters*, *setters*, el constructor y algún otro método como *equals*, *canEquals*, *hashCode* y *toString*. Y lo único que se ha hecho es anotar la clase con @Data y añadir su correspondiente import. Está claro que para que esto ocurra, nuestro IDE debe reconocer *lombok*, y por otro lado, nuestro proyecto debe importar la librería.

4. Instalando Lombok

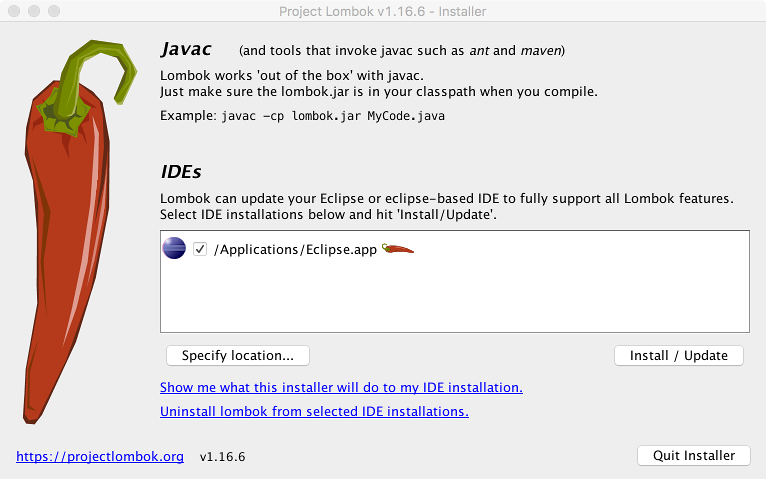
Desde la web del *proyecto Lombok* se puede [descargar](https://projectlombok.org/download.html) un jar, que debería poder instalarse con un simple doble click. En el caso de que esto no funcionara habría que recurrir a la línea de comandos y ejecutar:

Java



|  |  |
| --- | --- |
| 1 | java -jar lombok.jar |

El instalador intentará descubrir los IDEs que tienes instalados y preguntará en cuales quieres instalar Lombok. En este equipo me detectó el Eclipse. En caso de que no encuentre ninguno de los que soporta, se puede indicar dónde está instalado el IDE en el que queremos añadir lombok.

[](http://www.adictosaltrabajo.com/wp-content/uploads/2016/01/lombok_02.png)

También hay que añadir el JAR al proyecto donde lo queramos usar. Lo podemos añadir como dependencias de maven en el pom.xml

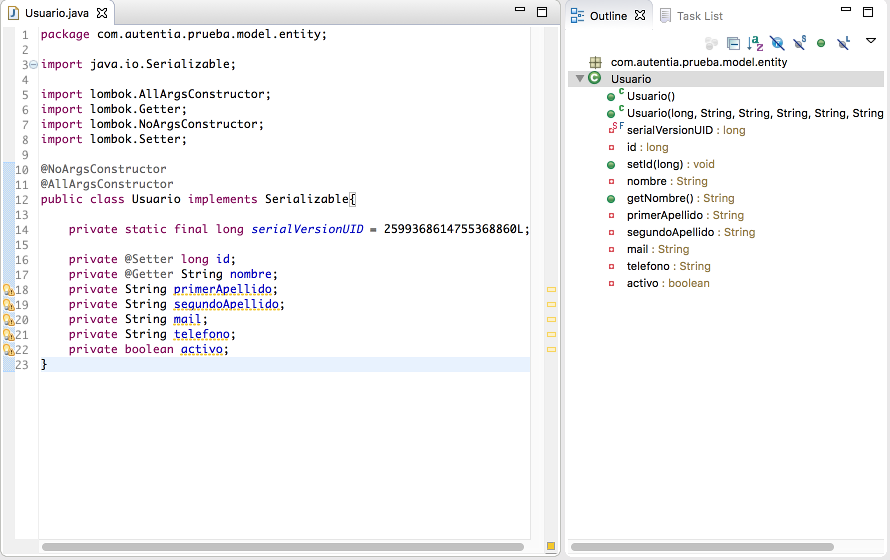
|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <dependency>       <groupId>org.projectlombok</groupId>       <artifactId>lombok</artifactId>       <version>1.16.6</version>       <scope>provided</scope>  </dependency> |

5. Algunas características de Lombok

Ya hemos visto lo que puede hacer la anotación @Data por nosotros. Pero hay muchas otras. Algunas de las más reseñables son:

* @Getter
* @Setter
* @NoArgsConstructor
* @AllArgsConstructor
* @Builder
* @Log

La mayoría son autoexplicativas y no merece la pena profundizar en ellas. Baste con un par de experimentaciones rápidas para darse cuenta de que lo que hacen es lo que dice la anotación. En este sentido, lombok es muy intuitivo.

[](http://www.adictosaltrabajo.com/wp-content/uploads/2016/01/lombok_03.png)

En el ejemplo anterior, hemos anotado la clase con @NoArgsConstructor y @AllArgsConstructor, y comprobamos que el resultado en el outline, es que tenemos dos constructores: uno sin argumentos y otro con todos los argumentos.

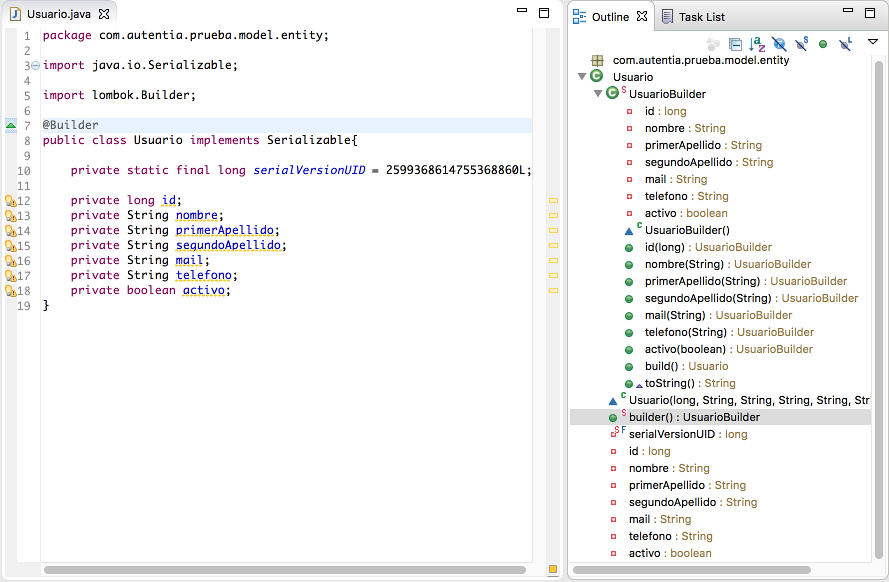
A nivel de propiedades, hemos anotado una con @Setter y otra con @Getter, y automáticamente nos genera dichos métodos para esas propiedades. Estas anotaciones se pueden usar a nivel de clase, y generaría esos métodos para todas las propiedades. Muy fácil.

.

6. Implementación del patrón Builder

Lombok también nos permite implementar el patrón Builder de forma efectiva en nuestro código, únicamente debemos anotar la clase con @Builder, y automáticamente nos creará el método público build y el objeto estático Builder con todos sus atributos y métodos.

Para quien no lo conozca, en el siguiente enlace se muestra el [funcionamiento del patrón Builder](https://jlordiales.wordpress.com/2012/12/13/the-builder-pattern-in-practice/) de forma muy didáctica.

[](http://www.adictosaltrabajo.com/wp-content/uploads/2016/01/lombok_04.png)

De esta forma siempre podremos hacer:

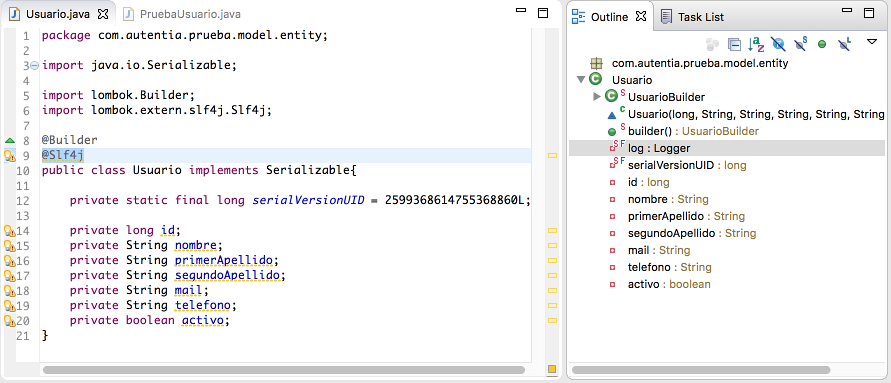
Java



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | private Usuario cliente                  = new Usuario.UsuarioBuilder()                                      .id(102)                                      .nombre("Alejandro")                                      .primerApellido("Magno")                                      .mail("correo@ejemplo.com")                                      .build(); |

7. Anotaciones para el tratamiento de logs

Hay seis posibles anotaciones de clase para el tratamientos de logs que crean la variable estática, privada y final log, dependiendo del sistema con el que se desee registrar las trazas.

[](http://www.adictosaltrabajo.com/wp-content/uploads/2016/01/lombok_05.png)

* @CommonsLog
* @Log
* @Log4j
* @Log4j2
* @Slf4j
* @XSlf4j

Se pueden ver explicadas con más detalle en la [~~página oficial~~](https://projectlombok.org/features/Log.html)

El resultado es que de forma sencilla y efectiva podemos tener control sobre nuestros POJOs con unas simples anotaciones, permitiéndonos aislarnos de la «morralla» habitual vinculada al tratamiento de este tipo de clases, que en el caso de modelos de datos con numerosos atributos, pueden ser muy tediosos de gestionar, aunque los IDEs actuales nos ayuden a ello.

8. Referencias

* Web Oficial de Lombok  
  [https://projectlombok.org](https://projectlombok.org/)
* Tutorial algo antañón  
  <http://jnb.ociweb.com/jnb/jnbJan2010.html>
* Explicación del patrón Builder  
  <https://jlordiales.wordpress.com/2012/12/13/the-builder-pattern-in-practice/>