# Usando Lombok en Java

Para usar Lombok en Java, necesitarás configurarlo en tu proyecto para simplificar tu código reduciendo el código boilerplate, como getters, setters y constructores. Aquí tienes cómo hacerlo paso a paso:

Primero, necesitas agregar Lombok como dependencia. Si estás usando Maven, incluye esto en tu `pom.xml`:

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.32</version> <!-- Verifica la última versión -->
    <scope>provided</scope>
</dependency>
```

Para Gradle, agrega esto a tu `build.gradle`:

```
provided 'org.projectlombok:lombok:1.18.32'
// También agrega esto para asegurarte de que el procesamiento de anotaciones funcione
annotationProcessor 'org.projectlombok:lombok:1.18.32'
```

A continuación, si estás usando un IDE como IntelliJ IDEA o Eclipse, necesitarás habilitar el procesamiento de anotaciones e instalar el plugin de Lombok: - **IntelliJ**: Ve a `File > Settings > Plugins`, busca “Lombok” , instálalo y reinicia. Luego asegúrate de que `Enable annotation processing` esté marcado bajo `Settings > Build, Execution, Deployment > Compiler > Annotation Processors`. - **Eclipse**: Descarga el JAR de Lombok desde el sitio oficial (projectlombok.org), ejecútalo con `java -jar lombok.jar` y sigue el instalador para apuntar a tu instalación de Eclipse.

Ahora, puedes comenzar a usar las anotaciones de Lombok en tu código Java. Aquí tienes un ejemplo simple:

```
import lombok.Getter;
import lombok.Setter;
import lombok.NoArgsConstructor;
import lombok.AllArgsConstructor;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class Person {
    private String name;
    private int age;
}
```

En este ejemplo: - `@Getter` y `@Setter` generan automáticamente los métodos getter y setter para `name` y `age`. - `@NoArgsConstructor` crea un constructor sin argumentos. - `@AllArgsConstructor` crea un constructor con todos los campos como parámetros.

Cuando compiles o ejecutes esto, Lombok procesa estas anotaciones y agrega el código boilerplate por ti. Puedes probarlo así:

```
public class Main {
    public static void main(String[] args) {
        Person person = new Person("Alice", 30);
        System.out.println(person.getName()); // Muestra: Alice
        person.setAge(31);
        System.out.println(person.getAge()); // Muestra: 31
    }
}
```

Algunas anotaciones populares que también podrías encontrar útiles: - `@Data`: Un atajo para `@Getter`, `@Setter`, `@ToString`, `@EqualsAndHashCode` y un constructor para campos requeridos (como los finales). - `@Builder`: Te da un patrón de constructor para la creación de objetos. - `@Slf4j`: Agrega un logger (por ejemplo, de SLF4J) a tu clase.

Por ejemplo, con `@Data` y `@Builder`:

```
import lombok.Data;
import lombok.Builder;

@Data
@Builder
public class User {
    private String username;
    private String email;
}

// Uso
public class Main {
    public static void main(String[] args) {
        User user = User.builder()
                        .username("john_doe")
                        .email("john@example.com")
                        .build();
        System.out.println(user); // Muestra: User(username=john_doe, email=john@example.com)
    }
}
```

Ten en cuenta: - Siempre verifica la última versión de Lombok en Maven Central o en el sitio oficial. - Si estás trabajando en equipo, asegúrate de que todos tengan Lombok configurado en su IDE para evitar confusiones con el código generado que no sea visible en el código fuente.