Utilisation de Lombok en Java

Pour utiliser Lombok en Java, vous devrez le configurer dans votre projet pour simplifier votre code en réduisant les éléments de code redondants, comme les getters, les setters et les constructeurs. Voici comment procéder étape par étape :

Tout d'abord, vous devez ajouter Lombok en tant que dépendance. Si vous utilisez Maven, incluez ceci dans votre `pom.xml` :

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.32</version> <!-- Vérifiez la dernière version -->
    <scope>provided</scope>
</dependency>
```

Pour Gradle, ajoutez ceci à votre `build.gradle` :

```
provided 'org.projectlombok:lombok:1.18.32'
// Ajoutez également ceci pour vous assurer que le traitement des annotations fonctionne
annotationProcessor 'org.projectlombok:lombok:1.18.32'
```

Ensuite, si vous utilisez un IDE comme IntelliJ IDEA ou Eclipse, vous devrez activer le traitement des annotations et installer le plugin Lombok : - **IntelliJ** : Allez dans `Fichier > Paramètres > Plugins`, recherchez "Lombok", installez-le et redémarrez. Ensuite, assurez-vous que `Activer le traitement des annotations` est coché sous `Paramètres > Build, Execution, Deployment > Compiler > Annotation Processors`. - **Eclipse** : Téléchargez le fichier JAR de Lombok depuis le site officiel (projectlombok.org), exécutez-le avec `java -jar lombok.jar`, et suivez l'installateur pour le pointer vers votre installation Eclipse.

Maintenant, vous pouvez commencer à utiliser les annotations Lombok dans votre code Java. Voici un exemple 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;
}
```

Dans cet exemple : - `@Getter` et `@Setter` génèrent automatiquement les méthodes getter et setter pour `name` et `age`. - `@NoArgsConstructor` crée un constructeur sans arguments. - `@AllArgsConstructor` crée un constructeur avec tous les champs en paramètres.

Lors de la compilation ou de l'exécution de ceci, Lombok traite ces annotations et ajoute le code redondant pour vous. Vous pouvez le tester comme ceci :

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

Quelques annotations populaires que vous pourriez également trouver utiles : - `@Data` : Un raccourci pour `@Getter`, `@Setter`, `@ToString`, `@EqualsAndHashCode`, et un constructeur pour les champs requis (comme les champs finaux). - `@Builder` : Vous donne un modèle de construction pour la création d'objets. - `@Slf4j` : Ajoute un journal (par exemple, de SLF4J) à votre classe.

Par exemple, avec `@Data` et `@Builder` :

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

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

// Utilisation
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); // Affiche : User(username=john_doe, email=john@example.com)
    }
}
```

N’oubliez pas : - Vérifiez toujours la dernière version de Lombok sur Maven Central ou sur le site officiel. - Si vous travaillez en équipe, assurez-vous que tout le monde a configuré Lombok dans leur IDE pour éviter toute confusion avec le code généré qui ne serait pas visible dans la source.