Project Lombok Annotations - Detailed Guide

@Getter / @Setter

Generates getter and setter methods for fields. You can apply it at the class level (to all fields) or at field level (to specific fields).

```
@Getter
@Setter
public class User {
    private String name;
    private int age;
}
```

@ToString

Generates a toString() method that includes all fields. Use @ToString.Exclude on fields you want to skip.

```
@ToString
public class User {
    private String name;
    @ToString.Exclude
    private String password;
}
```

@EqualsAndHashCode

Generates equals() and hashCode() methods. You can exclude specific fields using @EqualsAndHashCode.Exclude.

```
@EqualsAndHashCode
public class User {
    private String id;
    @EqualsAndHashCode.Exclude
    private String tempSession;
}
```

@NoArgsConstructor / @AllArgsConstructor / @RequiredArgsConstructor

@NoArgsConstructor: Generates a no-argument constructor.

@AllArgsConstructor: Constructor for all fields.

@RequiredArgsConstructor: Constructor for fields that are final or marked with @NonNull.

```
@RequiredArgsConstructor
public class Book {
    @NonNull private String title;
    private int year;
}
```

@Data

A shortcut that includes @Getter, @Setter, @ToString, @EqualsAndHashCode, and @RequiredArgsConstructor.

```
@Data
public class Employee {
    private String name;
    private int id;
}
```

Project Lombok Annotations - Detailed Guide

@Value

Used for immutable classes. All fields are private final and class is final. Generates constructor, getters, equals, hashCode, and toString.

```
@Value
public class Address {
    String city;
    String zipCode;
}
```

@Builder

Generates a builder pattern. Useful for constructing complex objects.

```
@Builder
public class Person {
    private String name;
    private int age;
}
```

@Singular

Used with @Builder to add individual elements to collections.

```
@Builder
public class Team {
    @Singular private List<String> members;
}
```

@SneakyThrows

Allows throwing checked exceptions without declaring them. Use with care.

```
@SneakyThrows
public void readFile() {
    throw new IOException("File error");
}
```

@Synchronized

Like Java's synchronized, but uses a private lock object for thread safety.

```
@Synchronized
public void safeMethod() {
    // thread-safe code
}
```

@Cleanup

Ensures that resources are closed automatically (like try-with-resources).

```
@Cleanup InputStream in = new FileInputStream("file.txt");
```

@Slf4j / @Log

Adds a logger instance based on the selected logging framework. Use it for easy logging.

```
@Slf4j
```

Project Lombok Annotations - Detailed Guide

```
public class LoggerExample {
    public void logMessage() {
        log.info("This is a log message");
    }
}
```