



Spring Framework 7

Beginner to Guru

Project Lombok Overview



What is Project Lombok

- Java is often criticized for having too much ceremonial code
- For example, getters and setters
- Project Lombok provides annotations which help eliminate the writing of ceremonial code
- `@Getter` - generates getters
- `@Setter` - generates setters
- Using Project Lombok will save you time and give you cleaner code





Project Lombok History

- Started by Reinier Zwitserloot - @surial on Twitter and Roel Spilker before 2009
- Why “Lombok”? Java is also an island in Indonesia. Lombok is the second island east of the Island Java.
- Lombok is also Indonesian for chilli.
- Hence tag line - “Spicing up your Java”





How Lombok Works

- Hooks in via the Annotation processor API
- The AST (raw source code) is passed to Lombok for code generation before java continues.
- Thus, produces properly compiled Java code in conjunction with the Java compiler
- NOTE: Code is generated and complied. No run-time performance penalty
- If you write an implantation of a method Project Lombok would generate, your code is used
 - Make it easy to override Lombok generated code
 - Example: custom setters





Project Lombok and IDEs

- Since compiled code is change, and source files are not, IDE's can get confused by this.
- More of an issue for IDEs several years old.
- Modern IDEs such as IntelliJ, Eclipse (and offshoots), Netbeans support Project Lombok
- Plugin Installation may be necessary





Project Lombok Features

- **val** - declares final local variable
- **var** - declares mutable local variable
- **@Getter**
 - Creates getter methods for all properties
- **@Setter**
 - Creates setter for all non-final properties





Project Lombok Features

- **@ToString**

- Generates String of classname, and each field separated by commas
- Optional parameter to include field names
- Optional parameter to include call to the super toString method





Project Lombok Features

- **@EqualsAndHashCode**
 - Generates implementations of 'equals(Object other)' and 'hashCode()'
 - By default will use all non-static, non-transient properties
 - Can optionally exclude specific properties





Project Lombok Features

- **@NoArgsConstructor**

- Generates no args constructor
- Will cause compiler error if there are final fields
- Can optionally force, which will initialize final fields with 0 / false / null





Project Lombok Features

- **@RequiredArgsConstructor**
 - Generates a constructor for all fields that are final or marked @NonNull
 - Constructor will throw a NullPointerException if any @NonNull fields are null





Project Lombok Features

- **@Data**
 - Generates typical boilerplate code for POJOs
 - Combines - @Getter, @Setter, @ToString, @EqualsAndHashCode, @RequiredArgsConstructor
 - No constructor is generated if constructors have been explicitly declared





Project Lombok Features

- **@Value**

- The immutable variant of @Data
- All fields are made private and final by default

- **@NonNull**

- Set on parameter of method or constructor and a NullPointerException will be thrown if parameter is null





Project Lombok Features

- **@Builder**

- Implements the 'builder' pattern for object creation
- Person.builder().name("Adam Savage").city("San Francisco").job("Mythbusters").job("Unchained Reaction").build();





Project Lombok Features

- **@SneakyThrows**
 - Throw checked exceptions without declaring in calling method's throws clause
- **@Syncronized**
 - A safer implementation of Java's synchronized





Project Lombok Features

- @Log
 - Creates a Java util logger
 - Java util loggers are awful
- @Slf4j
 - Creates a SLF4J logger.
 - Recommended - SLF4J is a generic logging facade
 - Spring Boot's default logger is LogBack



