Project Lombok

Denny Slover
Tech Lead
Masterpass Merchant Portal
MasterCard International

Overview

- Survey
    - Who has used Lombok before?
    - Who has heard of Lombok?
    - Who knows someone that has used it?
    - Who thinks it is a type of pepper?
- What is Lombok
- Benefits
    - Generates boilerplate code so you don’t have to
    - Makes the code consistent
- Lombok Plugin/Install for IDEs

Annotations

- @Getter
- @Setter
- @ToString
- @EqualsAndHashCode
- @RequiredArgsConstructor
- @Data
- @Value
- @NoArgsConstructor
- @AllArgsConstructor
- @Log
- @Builder
- @Singular

@Getter & @Setter

```
public class GetterSetterExample{
@Getter
@Setter
private int age = 10;
}
```

```
public class GetterSetterExample{
private int age = 10;

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}
```

@Getter & @Setter

- Automatically creates isFieldName for boolean attributes.
- Annotations can be field or class level.
- Generated method names are created with the field name capitalised and then get/set/is prefixed.
  boolean getters will be names isFieldName but if (java.lang.) Boolean is used, method name will be getFieldName.
- Lombok is very polite
  If a method is created with the same name as the generated method, Lombok will suppress code generation for that field
- enum
  @Getter can be used. @Setter can not.
- Annotation options
  - AccessLevel
    - public (default)
      @Getter(AccessLevel.***PUBLIC***)
      public int getAge() { ..}
    - private
      @Getter(AccessLevel.***PRIVATE***)
      private int getAge() { ..}
    - protected
      @Getter(AccessLevel.***PROTECTED***)
      protected int getAge() { .. }

@ToString

- Prints class name(fully qualified) and all fields, in order, separated by commas
- Caution on bidirectional relationships
  - Use exclude/of as needed
  - @ToString(of = {"userName", “firstName"})
  - @ToString(exclude = {"userName", "firstName"})

@EqualsAndHashCode

- @EqualsAndHashCode(callSuper=true)
 public static class Square extends Shape {
 private final int width, height;
 }
- Adds 3 methods
  - equals(Object o)
  - canEqual(Object o)
  - hasCode()
- Use exclude/of as needed - Usage is the same as @ToString annotation

Generated Methods

```java
public boolean equals(Object o) {
    if (o == this) return true;
    if (!(o instanceof EqualsHashCodeUser)) return false;
    final EqualsHashCodeUser other = (EqualsHashCodeUser) o;
    if (!other.canEqual((Object) this)) return false;
    final Object this$userId = this.getUserId();
    final Object other$userId = other.getUserId();
    if (this$userId == null ? other$userId != null : !this$userId.equals(other$userId)) return false;
    final Object this$firstName = this.getFirstName();
    final Object other$firstName = other.getFirstName();
    if (this$firstName == null ? other$firstName != null : !this$firstName.equals(other$firstName)) return false;
    final Object this$lastName = this.getLastName();
    final Object other$lastName = other.getLastName();
    if (this$lastName == null ? other$lastName != null : !this$lastName.equals(other$lastName)) return false;
    final Object this$userName = this.getUserName();
    final Object other$userName = other.getUserName();
    if (this$userName == null ? other$userName != null : !this$userName.equals(other$userName)) return false;
    return true;
}

public int hashCode() {
    final int PRIME = 59;
    int result = 1;
    final Object $userId = this.getUserId();
    result = result * PRIME + ($userId == null ? 43 : $userId.hashCode());
    final Object $firstName = this.getFirstName();
    result = result * PRIME + ($firstName == null ? 43 : $firstName.hashCode());
    final Object $lastName = this.getLastName();
    result = result * PRIME + ($lastName == null ? 43 : $lastName.hashCode());
    final Object $userName = this.getUserName();
    result = result * PRIME + ($userName == null ? 43 : $userName.hashCode());
    return result;
}

protected boolean canEqual(Object other) {
    return other instanceof EqualsHashCodeUser;
}
```

@RequiredArgsConstructor

- Creates a constructor with parameters for all required fields
  - All non-initialized final fields
  - All non-initialized fields that are marked as @NonNull
- Order in the constructor is the same order as the fields appear in the class

@Data

- One annotation to rule them all
- Wraps all these annotations into one
  - @Getter
  - @Setter
  - @ToString
  - @EqualAndHashCode
  - @RequiredArgsConstructor
- No individual annotation configuration.
  - ~~@Data(of = {"userName", “firstName”})~~ This will not work!
  - This will work….
    @Data
    @ToString(of = {"userName", “firstName"})

@Value

- The immutable @Data annotation!
- All fields are made private and final by default
- No setters generated

Further Constructor Annotations

- In addition to the @RequiredArgsConstructor, two other constructor annotations
  - @NoArgsConstructor
  - @AllArgsConstructor

@Log + Friends

- @CommonsLog
  Creates private static final org.apache.commons.logging.Log log = org.apache.commons.logging.LogFactory.getLog(LogExample.class);

- @JBossLog
  Creates private static final org.jboss.logging.Logger log = org.jboss.logging.Logger.getLogger(LogExample.class);

- @Log
  Creates private static final java.util.logging.Logger log = java.util.logging.Logger.getLogger(LogExample.class.getName());

- @Log4j
  Creates private static final org.apache.log4j.Logger log = org.apache.log4j.Logger.getLogger(LogExample.class);

- @Log4j2
  Creates private static final org.apache.logging.log4j.Logger log = org.apache.logging.log4j.LogManager.getLogger(LogExample.class);

- **@Slf4j**
  Creates private static final org.slf4j.Logger log = org.slf4j.LoggerFactory.getLogger(LogExample.class);

- @XSlf4j
  Creates private static final org.slf4j.ext.XLogger log = org.slf4j.ext.XLoggerFactory.getXLogger(LogExample.class);

- Note, either annotation will result in log field.

@Builder
Builder API's made easy!

- Making a better way to create objects.
- @Singular - Collection? I can handle that for you!

-
  ```
  @Builder
  public class Person {
      private String name;
      private String city;
      private List<String> job;
  }
  ```

-
  ```
  Person
  .builder()
  .name("Adam Savage")
  .city(“San Francisco”)
  .job("Mythbusters")
  .job("Unchained Reaction")
  .build();
  ```

@NonNull

- @NonNull
  - Field level
    - Adds null check to any Lombok generated methods (setters and constructors)
    - Throws a NullPointerException if null
  - Parameter level
    - Adds null check to in the method
    - Throws a NullPointerException if null
  - public void nullExample(@NonNull String str) {… }

-
  ```
  public void nullExample(String str) {
  if (str == null) {
        throw new NullPointerException("str");
      }
  ```

The Rest of the Framework

- @Cleanup
  - Automatically cleaning up and close your variables when leaving scope
- val
  - val str = new String();
    String str = new String();
  - Just like JavaScript!
  - Used only on method local variables (not class fields)

Questions?

- Questions? Comments?
- Who wants to start using Lombok now?
- Further resources
    - https://projectlombok.org/index.html
    - http://jnb.ociweb.com/jnb/jnbJan2010.html
    - http://zeroturnaround.com/rebellabs/why-the-lazy-coder-in-you-would-enjoy-project-lombok/
    - https://blog.codecentric.de/en/2015/11/less-but-more-expressive-code-with-project-lombok/