부록 B. Lombok 사용법

롬복은 코드 작성양을 줄여주는 기능을 제공한다. 롬복은 이클립스 플러그인이다. 롬복은 ~.java 파일에 설정한 롬복 애노테이션을 컴파일 할 때 실제 코드로 대체해 주는 역할을 수행한다.

1. 다운로드

```
Google >> lombok download >> Site >> download
```

2. 설치

STS를 종료한 상태에서 다음 명령을 실행한다.

```
java -jar lombok-1.18.2.jar

Installer 뷰 >> Specify Location >> 사용하는 STS.exe 선택 >> Install 클릭
```

3. 프로젝트에서 사용하기

프로젝트의 디펜던시에 롬복을 추가한다. 스프링 부트 프로젝트가 아니고 Spring Legacy Project 인 경우, version 항목도 설정해야 한다.

pom.xml

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <scope>compile</scope>
</dependency>
```

Step 1. 롬복을 사용하지 않을 때에 모습

NonLombokModel.java

```
package com.example.demo.lombok.step1;

// @Data :
// @Getter, @Setter, @RequiredArgsConstructor,
// @ToString, @EqualsAndHashCode 애노테이션들을 붙이는 대신 사용한다.
```

```
public class NonLombokModel {

    private String name;
    private int age;
    private String address;

    // @NoArgsConstructor
    public NonLombokModel() {}

    // @AllArgsConstructor
    // 또는
    // @RequiredArgsConstructor
    // @NonNull String name;
    // @NonNull Integer age;
    // @NonNull String address;
    public NonLombokModel(String name, int age, String address) {
        super();
        this.name = name;
        this.age = age;
        this.address = address;
    }

    // @Builder
    public static NonLombokModelBuilder builder() {
        NonLombokModel lombokModel = new NonLombokModel();
        return lombokModel.new NonLombokModelBuilder();
    }

    // @Builder
    class NonLombokModelBuilder {
        public NonLombokModelBuilder name(String name) {
            NonLombokModel.this.name = name;
            return this;
        }

        public NonLombokModelBuilder age(int age) {
            NonLombokModel.this.age = age;
            return this;
        }

        public NonLombokModelBuilder address(String address) {
            NonLombokModel.this.address = address;
            return this;
        }

        public NonLombokModel build() {
            return NonLombokModel.this;
        }
    }

    // @ToString
    @Override
    public String toString() {
        return "NonLombokModel [name=" + name + ", age=" + age + ", address=" + address + "]";
    }

    // @Getter
    public String getName() {
        return name;
    }

    // @Setter
    public void setName(String name) {
        this.name = name;
```

```
    }

    // @Getter
    public int getAge() {
        return age;
    }

    // @Setter
    public void setAge(int age) {
        this.age = age;
    }

    // @Getter
    public String getAddress() {
        return address;
    }

    // @Setter
    public void setAddress(String address) {
        this.address = address;
    }

    // @EqualsAndHashCode
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((address == null) ? 0 : address.hashCode());
        result = prime * result + age;
        result = prime * result + ((name == null) ? 0 : name.hashCode());
        return result;
    }

    // @EqualsAndHashCode
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
            return false;
        NonLombokModel other = (NonLombokModel) obj;
        if (address == null) {
            if (other.address != null)
                return false;
        } else if (!address.equals(other.address))
            return false;
        if (age != other.age)
            return false;
        if (name == null) {
            if (other.name != null)
                return false;
        } else if (!name.equals(other.name))
            return false;
        return true;
    }

}
```

NonLombokModelTest.java

```
package com.example.demo.lombok.step1;

public class NonLombokModelTest {

    public static void main(String[] args) {
        NonLombokModel model1 = new NonLombokModel("Tom", 56, "NY");

        System.out.println(model1);
        System.out.println(model1.hashCode());

        NonLombokModel model2 = NonLombokModel.builder().name("Tom").age(56).address("NY").build();

        System.out.println(model2);
        System.out.println(model2.hashCode());

        // false : 동일비교
        System.out.println(model1 == model2);

        // true : 동등비교
        System.out.println(model1.equals(model2));
    }

}
```

Step 2. Domain 클래스를 위한 롬복 애노테이션

- @Getter: getter 메소드 코드를 제너레이트 한다.
- @Setter: setter 메소드 코드를 제너레이트 한다.
- @RequiredArgsConstructor: @NonNull이 설정된 멤버변수를 파라미터로 받는 생성자를 만든다.
- @ToString: toString() 메소드를 재 정의 한다.
- @EqualsAndHashCode: hashCode(), equals() 메소드를 재 정의 한다.
- @Data: 위 애노테이션들을 설정하는 대신 이 애노테이션을 사용할 수 있다. 설정 결과는 동일한다.
- @Builder: 새 객체를 빌더로직을 통해서 만들 수 있는 코드를 제너레이트 한다.
- @NoArgsConstructor: 디폴트 생성자 코드를 제너레이트 한다.
- @AllArgsConstructor: 멤버변수들 모두를 파라미터로 받는 생성자 코드를 제너레이트 한다.

LombokModel.java

```
package com.example.demo.lombok.step2;

import lombok.Builder;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
import lombok.ToString;
```

```
@Getter
@Setter
@RequiredArgsConstructor
@ToString
@EqualsAndHashCode
// @Data
@Builder
public class LombokModel {
    private @NonNull String name;
    private @NonNull Integer age;
    private @NonNull String address;
}
```

LombokModelTest.java

```
package com.example.demo.lombok.step2;

public class LombokModelTest {

    public static void main(String[] args) {
        // @RequiredArgsConstructor
        // @NonNull String name;
        // @NonNull Integer age;
        // @NonNull String address;
        LombokModel model1 = new LombokModel("Tom", 56, "NY");

        // @ToString
        System.out.println(model1);
        // @EqualsAndHashCode
        System.out.println(model1.hashCode());

        // @Builder : 빌더를 사용하여 새 객체를 생성한다.
        LombokModel model2 = LombokModel.builder().name("Tom").age(56).address("NY").build();

        // @ToString
        System.out.println(model2);
        // @EqualsAndHashCode
        System.out.println(model2.hashCode());

        // false : 동일비교
        System.out.println(model1 == model2);

        // @EqualsAndHashCode
        // true : 동등비교
        System.out.println(model1.equals(model2));
    }

}
```

Step 3. @Accessors

LoginResult.java

```
package com.example.demo.lombok.step3;

import lombok.Getter;
```

```
import lombok.NonNull;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
import lombok.ToString;
import lombok.experimental.Accessors;

@RequiredArgsConstructor
@Accessors(fluent = true, chain = true)
@Getter
@Setter
@ToString(exclude = { "id" }) // 멤버변수 id는 제외한다.
public class LoginResult {
    private String id;
    private @NonNull String firstName;
    private @NonNull String lastName;
}
```

LoginResultTest.java

```
package com.example.demo.lombok.step3;

public class LoginResultTest {

    public static void main(String[] args) {
        // @RequiredArgsConstructor : @NonNull이 있는 멤버변수를 생성자로 받는다.
        // @NonNull String firstName;
        // @NonNull String lastName;
        LoginResult loginResult = new LoginResult("Tom", "Cruise");

        // @Accessors(fluent = true) : 멤버변수명을 그대로 메소드명으로 사용한다.
        System.out.println(loginResult.id());
        System.out.println(loginResult.firstName());
        System.out.println(loginResult.lastName());

        // @Accessors(chain = true) : setter 메소드가 this를 리턴하여 메소드 체이닝 기법을 사용할 수 있다.
        loginResult.id("tom.cruise@actor.org").firstName("TOM").lastName("CRUISE");

        // @ToString(exclude = { "id" }) : 멤버변수 id는 제외한다.
        System.out.println(loginResult);
    }

}
```

Step 4. @Slf4j, @Cleanup, @SneakyThrows

example.txt

```
-----------
I love you!
===========
```

CheckedExceptionExample.java

```java
package com.example.demo.lombok.step4;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.nio.charset.UnsupportedCharsetException;
import java.util.stream.Collectors;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import lombok.Cleanup;
import lombok.SneakyThrows;
import lombok.extern.slf4j.Slf4j;

@Slf4j
public class CheckedExceptionExample {

    // @Slf4j 애노테이션으로 대신할 수 있다.
//    private static Logger log = LoggerFactory.getLogger(CheckedExceptionExample.class);

    public static void main(String[] args) {
        CheckedExceptionExample demo = new CheckedExceptionExample();

        String file = "example.txt";

        String data = demo.resourceAsString(file);
        System.out.println(data);

        String dataAnother = demo.resourceAsStringAnother(file);
        System.out.println(dataAnother);
    }

    public String resourceAsString(String file) {
        StackTraceElement[] elements = new Throwable().getStackTrace();
        log.info("Callee : {}", elements[0].getClassName() + "." + elements[0].getMethodName());

        // @Cleanup("close") : InputStream 객체를 사용하고 난 후, close() 메소드가
        // 호출되게 만든다.
        try (InputStream is = this.getClass().getResourceAsStream(file)) {
            BufferedReader br = new BufferedReader(new InputStreamReader(is, "UTF-8"));
            return br.lines().collect(Collectors.joining("\n"));
        }
        // @SneakyThrows : 체크드 익셉션을 언체크드 익셉션으로 변경하여 던진다.
        catch (IOException | UnsupportedCharsetException ex) {
            throw new RuntimeException(ex);
        }
    }

    @SneakyThrows
    public String resourceAsStringAnother(String file) {
        StackTraceElement[] elements = new Throwable().getStackTrace();
        log.info("Caller : {}", elements[0].getClassName() + "." + elements[0].getMethodName());

        @Cleanup("close") InputStream is = this.getClass().getResourceAsStream(file);
        BufferedReader br = new BufferedReader(new InputStreamReader(is, "UTF-8"));
        return br.lines().collect(Collectors.joining("\n"));
    }

}
```