롬복(lombok)소개

- 자바개발시 모델객체를 생성할 때 setter/getter/toString/hashCode/equals 메소드를 만드는데 이럴 경우 클래스 파일의 소스가 길어지고 복잡해지는데, 이를 해결하기 위해 롬복(Lombok)을 사용한다.
- 클래스 안에 있는 필드에 대해 Getter, Setter의 생성이나, toString(), equals(), hashCode() 메서드, 생성자를 자동으로 생성 해준다.
- 설치
  - ✓ http://projectlombok.org/download.html 에서 jar 파일을 다운로드 후 실행(더블클릭해서 실행 안되면 javaw –jar lombok.jar로 실행하자)
  - ✓ maven의 설정 파일에 의존성 추가

```
<dependency>groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.16.6</version></dependency>
```

  - ✓ Gradle의 설정 파일에 의존성 추가

```
Compile('org.projectlombok:lombok:1.16.6')
```

스프링 부트를 사용한다면 프로젝트 생성시 두번째 화면에서 Core -> Lombok을 선택하면 자동으로 라이브러리가 추가된다.

- 어노테이션

@Getter : Getter 메소드를 생성해 준다.
@Setter : Setter 메소드를 생성해 준다.
@ToString : 클래스의 필드를 확인해서 toString 메소드를 적절히 만들어 준다.

```
@Entity
@Table(name="sawon_hobby")
@Getter @Setter
@ToString(exclude={"sawon", "hobby"})
@NoArgsConstructor  //아규먼트 없는 생성자를 만듬
public class SawonHobby {
    ……
}
```

@EqualsAndHashCode : equlas와 hashcode 메소드를 만들어 준다.
@Data : 클래스안의 모든 private 필드에 대해 @Getter와 @Setter를 적용하여 Getter/Setter를 만

들어주고 클래스내에 @ToString 과 @EqualsAndHashCode를 적용시켜 메소드를 오버라이드 해주며 @RequiredArgsConstructor를 지정하여 생성자를 만들어 준다.

val : 로컬변수에 사용되며 final을 사용한것과 비슷한 효과를 내며 형식을 자동 유추하여 타입을 따로 쓰지 않고 변수를 선언할 수 있다.

@NoArgsConstructor, @RequiredArgsConstructor, @AllArgsConstructor : 생성자를 자동으로 만들어준다. 특히 자바에서는 파라미터 있는 생성자를 만들어 놓은 경우 기본생성자(인자없는 생성자)를 자동으로 만들지 않으므로 @NoArgsConstructor를 사용하면 유용하다.

@RequiredArgsConstructor, All non-initialized final fields get a parameter, as well as any fields that are marked as @NonNull that aren't initialized where they are declared. For those fields marked with @NonNull, an explicit null check is also generated.

@NoArgsConstructor creates an empty constructor.

@AllArgsConstructor creates a constructor for all fields

@Delegate : 한 클래스에 다른 클래스의 메소드를 위임하여 생성하게 한다. 예문을 보자.

```
public class Test {
    @Delegate(types=Set.class)    //Set인터페이스의 메소드가 Test 클래스에 생긴다.
                                  //그리고 Test에서 Set의 메소드를 호출하도록 만들어준다.
    private final Set<String> emps = new HashSet<String>();
}
다음과 코드가 비슷하다.
public class DelegationTest {
    private final Set<String> emps =
                            new HashSet<String>();
    public boolean add(final String item) {
        return this.emps.add(item);
    }

    public boolean remove(final String item) {
        return this.emps.remove(item);
    }
    ......
}
```

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

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