어노테이션

Annotation

- 클래스, 메소드, 변수등에 추가적인 데이터(Meta Data)를 붙이는 방식
- XML을 대용하는 설정 파일로 사용
- 동적으로 클래스나 메소드에 필요한 메타 데이터를 이용할 수 있는 새로운 방식
- Annotation방식을 사용하는 기술 / 프레임웍들
  - ✓ Spring
  - ✓ Struts2
  - ✓ Mybatis

생성 규칙

- **interface** 키워드와 함께 @로 시작하고 어노테이션 이름을 명시
- 메서드에 파라미터를 선언 할 수 없음
- 메서드에 **throws**를 선언 할 수 없음
- 메서드의 반환타입으로 다음 중 하나를 선언
  - ✓ **primitives**
  - ✓ **String**
  - ✓ **Class**
  - ✓ **enum**
  - ✓ **array of the above types**

Java Built-In Annotation

- **java.lang** 패키지의 메타어노테이션(사용자 정의 어노테이션 작성)
  - ✓ **@Target**
  - ✓ **@Retention**
  - ✓ **@Documented**
  - ✓ **@Inherited**
- **java.lang** 패키지의 기본 어노테이션
  - ✓ **@Override**
  - ✓ **@Deprecated**
  - ✓ **@SuppressWarnings**

Java Meta Annotation(Target)

- @Target(ElementType.FIELD)
- The Target meta-annotation is used to specify where the annotation is to be applied.
  - ✓ FIELD
  - ✓ TYPE (Class, interface or enum definition)
  - ✓ METHOD
  - ✓ PARAMETER
  - ✓ CONSTRUCTOR
  - ✓ LOCAL_VARIABLE
  - ✓ ANNOTATION_TYPE
  - ✓ PACKAGE

Java Meta Annotation(Retention)

- @Retention(RetentionPolicy.RUNTIME)
- The Retention meta-annotation is used to specify how long an annotation is retained
  - ✓ SOURCE : This annotation information is only retained in the source code and is not recorded in the generated class file.
  - ✓ CLASS : This annotation is recorded in the class file by the compiler, but need not be retained by the virtual machine at runtime.
    This is the default if @Retention is not specified.
  - ✓ RUNTIME: Annotations are recorded in the class file by the compiler and retained by the virtual machine at runtime and can be read reflectively.

Annotation Types

- Marker
- Single-value
- Full-value or multi-value

Marker

- 어노테이션 선언만 있는 형태.
- 작성 :

```
public @interface MyAnnotation {
}
```

- 사용:

```
@MyAnnotation
public void myMethod() {
}
```

Single-value

- 어노테이션 선언시 하나의 데이터만을 표현한 형태.
- 작성 :

```
public @interface MyAnnotation {
    String value();
}
```

- 사용:

```
@MyAnnotation (value="test")
public void myMethod() {
}
```

Full-value

- 어노테이션 선언시 여러 개의 멤버를 표현한 형태.
- 작성 :

```
public @interface MyAnnotation {
    String value();
    String name();
}
```

- 사용:

```
@MyAnnotation (value="test", name="mrson")
public void myMethod() {
}
```