오라클자바커뮤니티교육센터
OJC ojcedu.com
구로디지털단지역 4분거리

자바 어노테이션(Java Annotation) 이란, 커스텀 어노테이션(Custom Annotation)

1. 자바 애노테이션이란(Java Annotation)?

- 자바 5 에서 부터 지원됨, @를 이용한 주석, 자바코드에 주석을 달아 특별한 의미를 부여한 것
- 메타데이터(실제데이터가 아닌 Data 를 위한 데이터) 기능을 자바로 가지고 와서 사용하는 것
- 컴파일러가 특정 오류를 억제하도록 지시하는 것과 같이 프로그램 코드의 일부가 아닌 프로그램에 관한 데이터를 제공, 코드에 정보를 추가하는 정형화된 방법.

2. Annotation 의 기능

메타데이터를 프로그램 요소(클래스, 인터페이스, 메소드 등)에 연결하는 방법을 제시
해당 엘리먼트에 대해 생성된 자바 바이트코드를 변경하지 추가 수식자 (modifier)라 할 수 있음
코드에 명시적 프로그래밍을 줄이고 좀더 많은 선언문을 제공한다.
즉 프로그램의 의미적인 부분에 직접 영향을 주지 않는다.

3. Annotation 사용

Annotation 은 @로 시작하고 그 뒤에 이을 붙여서 사용한다.
자바 5 에서는 세가지 Annotation 을 제공하며 직접 정의하는 것도 가능

@Override : 메소드를 재정의(OVerriding)할 것임을 컴파일러에게 알려준다.
메소드 선언하며 지정한 메소드가 부모클래스로부터 오버라이드 된 메소드임을 명시함
@Deprecated : 클래스, 메소드, 필드 등에 선언하며 지정한 요소가 더이상 사용되지 않음을 의미함(가급적 사용을 자제해달라는 의미)
@Deprecated Annotation 이 붙은 메소드등은 더이상 사용하지 말라고 컴파일러에게 알리며
컴파일러는 이러한 비추천메소드를 사용할 때 마다 경고를 표시한다.

@SuppressWarning : 클래스, 메소드, 필드의 선언, 컴파일러의 경고를 제거(이 부분에 대해서 경고문을 출력하지 말라는 의미)

4. 커스텀 Annotation

커스텀 Annotation 을 만들때 사용되는 4 개의 추가적인 메타 Annotation

@Target : 어노테이션(Annotation)을 적용할 대상을 지정함
value 값으로는 ElementType 에 enum 상수인 다음의 값들을 사용할수있다.
즉 어디에 어노테이션을 넣을 수 있는지를 서술하는데 field, method, class 가 정의된 곳에 어노테이션을 넣을 수 있다.

CONSTRUCTOR : 생성자
FIELD : enum 상수를 포함한 필드
LOCAL_VARIABLE : 지역 변수
METHOD : 메소드
PACKAGE : 패키지
PARAMETER : 파라미터
TYPE : Class, Interface, Enumeration 에 어노테이션을 적용

@Retention : 어노테이션이 얼마나 오랫동안 유지되는지에 대해, JVM 이 어떻게 사용자 어노테이션을 다루어야 하는지를 서술
Annotation 정보가 언제까지 유지될지 지정, 어노테이션 정보가 보관되는 기간
value 값으로는 RententionPolicy 에 enum 상수인 다음의 값들을 사용할수있음

SOURCE : 어노테이션이 컴파일 타임시 버려진다는 것을 의미, 클래스파일은 애노테이션을 가지지 못함
CLASS : 어노테이션이 생성된 클래스 파일에서 나타날 것이라는 것을 의미, 런타임시에는 이 어노테이션을 이용하지 못함
RUNTIME : 소스, 클래스, 실행시에 사용됨

@Documented : Annotation 을 JavaDoc 에 포함함
@Inherited : 부모 Annotation 을 상속받음, 기본적으로 어노테이션은 상속되지 않는다.
따라서 상속을 원한다면, 어노테이션을 @Inherited 해야 하며, 사용자 어노테이션에 놓여야 하며 클래스에만 효과가 있다.

[예제]

1.OnjInfo.java

```
import java.lang.annotation.ElementType;
import java.lang.annotation.Inherited;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(ElementType.METHOD)
@Inherited
@Retention(RetentionPolicy.RUNTIME)
public @interface OnjInfo{
	String name() default "oraclejava community";
	String year() default "2007";
	String desc() default "개발자실무교육";
}
```

2. Test.java

```
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.List;

public class Test {

	public static void main(String[] args) {
	}

	@Override
	@OnjInfo(name = "onj", year = "2014", desc = "onjoraclejava")
	public String toString() {
		return "Overriding toString method";
	}

	@Deprecated
	@OnjInfo(name = "oraclejava")
	public static void oldMethod() {
		System.out.println("old method(oraclejava),사용중지.");
	}
```

```
    @OnjInfo(name = "onjoraclejava", year="2007")
    public static void newMethod() {
        System.out.println("new method(onjoraclejava),OK.");
    }

    @SuppressWarnings({"unchecked", "deprecation" })
    @OnjInfo(name = "onj", year="2200")
    public static void genericsTest() throws FileNotFoundException {
        List l = new ArrayList();
        l.add("oraclejava");
        oldMethod();
    }

}
```

3. AnnoParse.java

```
import java.lang.annotation.Annotation;
import java.lang.reflect.Method;

public class AnnoParse {

    public static void main(String[] args) {
        try {
            for (Method method : AnnoParse.class
                    .getClassLoader()
                    .loadClass(("Test"))
                    .getMethods()) {
                // checks if MethodInfo annotation is present for the method
                if (method
                        .isAnnotationPresent(OnjInfo.class)) {
                    try {
                        // iterates all the annotations available in the method
                        for (Annotation anno :
method.getDeclaredAnnotations()) {
                            System.out.println("Annotation in Method '"
                                    + method + "' : " + anno);
                        }
                        OnjInfo methodAnno = method
                                .getAnnotation(OnjInfo.class);
```

```
                        if (methodAnno.year() == "2200") {
                            System.out.println("forever!! oraclejava, onj"+
method);
                        }

                    } catch (Throwable ex) {
                        ex.printStackTrace();
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

[결과]

```
Annotation in Method 'public static void Test.newMethod()' :
@OnjInfo(desc=개발자실무교육, name=onjoraclejava, year=2007)
Annotation in Method 'public java.lang.String Test.toString()' :
@OnjInfo(desc=onjoraclejava, name=onj, year=2014)
Annotation in Method 'public static void Test.oldMethod()' :
@java.lang.Deprecated()
Annotation in Method 'public static void Test.oldMethod()' :
@OnjInfo(desc=개발자실무교육, name=oraclejava, year=2007)
Annotation in Method 'public static void Test.genericsTest() throws
java.io.FileNotFoundException' : @OnjInfo(desc=개발자실무교육, name=onj,
year=2200)
```

실무프로그래머 전문교육
(오라클,SQL,자바,스프링프레임워크,닷넷,안드로이드,웹퍼블리싱)
오라클자바커뮤니티교육센터(100%환급, 개인부담 0~20%) ***http://ojcedu.com***