1、课程名称：深入 Annotation

MLDN
JAVA培训先锋

JAVA 应用开发详解

第16章：Annotation
—— 深入Annotation

MLDN 软件教学研发部

2、本课程预计讲解的知识点

本章目标

- 掌握@Target注释的作用
- 掌握@Documented注释的作用
- 掌握@Inherited注释的作用

E-MAIL：mldnqa@163.com www.MLDNJAVA.cn

3、具体内容

之前定义的 Annotation，如果没有明确的说明可以在任意的位置上使用。

```
@MyDefaultAnnotationReflect(key="MLDN",value="www.mldnjava.cn")
public class SimpleBean{
	@MyDefaultAnnotationReflect(key="MLDN",value="www.mldnjava.cn")
	public String toString(){
		return "Hello LiXingHua!!!" ;
	}
```

```
};
```

如果现在需要指定其使用的范围的话，则就必须使用@Target 注释。

```
@Documented
@Retention(value=RUNTIME)
@Target(value=ANNOTATION_TYPE)
public @interface Target
```

ElementType的保存范围

No.	范围	描述
1	public static final ElementType ANNOTATION_TYPE	只能用在注释声明上
2	public static final ElementType CONSTRUCTOR	只能用在构造方法声明上
3	public static final ElementType FIELD	只能用在字段声明（包括枚举常量）上
4	public static final ElementType LOCAL_VARIABLE	只能用在局部变量声明上
5	public static final ElementType METHOD	只能用在方法的声明上
6	public static final ElementType PACKAGE	只能用在包的声明上
7	public static final ElementType PARAMETER	只能用在参数的声明上
8	public static final ElementType TYPE	只能用在类、接口、枚举类型上

E-MAIL：mldnqa@163.com www.MLDNJAVA.cn

现在定义一个 Annotation，只能在类上使用，类型是 TYPE

```
import java.lang.annotation.Target ;
import java.lang.annotation.ElementType ;
import java.lang.annotation.Retention ;
import java.lang.annotation.RetentionPolicy ;
@Target(ElementType.TYPE)		// 此注释只能用在类上
@Retention(value=RetentionPolicy.RUNTIME)
public @interface MyTargetAnntation{
	public String key() default "LXH" ;
```

```
	public String value() default "李兴华" ;
}
```

此时在 SimpleBean 的类及方法的声明上使用此 Annotation

```
@MyTargetAnntation(key="MLDN",value="www.mldnjava.cn")
public class SimpleBean{
	@MyTargetAnntation(key="MLDN",value="www.mldnjava.cn")
	public String toString(){
		return "Hello LiXingHua!!!" ;
	}
};
```

如果现在希望一个 Annotation 可以在类及方法上同时使用的话，则必须设置多个范围：

```
import java.lang.annotation.Target ;
import java.lang.annotation.ElementType ;
import java.lang.annotation.Retention ;
import java.lang.annotation.RetentionPolicy ;
@Target({ElementType.TYPE,ElementType.METHOD})		// 此注释只能用在类上
@Retention(value=RetentionPolicy.RUNTIME)
public @interface MyTargetAnntation{
	public String key() default "LXH" ;
	public String value() default "李兴华" ;
}
```

@Documented，可以用在任何的 Annotation 上使用。所有的 Annotation 默认情况下都是使用@Documented 进行注释的，而且在生成 javadoc 的时候可以通过@Documented 设置一些说明信息。

```
import java.lang.annotation.Documented ;
@Documented
public @interface MyDocumentedAnntation{
	public String key() default "LXH" ;
	public String value() default "李兴华" ;
}
```

成功之后，在使用此 Annotation 的时候就可以增加一些信息上去。

```
@MyDocumentedAnntation(key="MLDN",value="www.mldnjava.cn")
public class SimpleBeanDocumented{
	/**
	 * 此方法在对象输出时调用，返回对象信息
	 */
	@MyDocumentedAnntation(key="MLDN",value="www.mldnjava.cn")
	public String toString(){
```

```
		return "Hello LiXingHua!!!" ;
	}
};
```

之后通过 javadoc 命令，生成 java doc 文档。

```
D:\otherann>javadoc -d doc SimpleBeanDocumented.java
正在创建目标目录： "doc\"
正在装入源文件 SimpleBeanDocumented.java...
正在构造 Javadoc 信息...
标准 Doclet 版本 1.6.0_11
正在构建所有软件包和类的树...
正在生成 doc\SimpleBeanDocumented.html...
正在生成 doc\package-frame.html...
正在生成 doc\package-summary.html...
正在生成 doc\package-tree.html...
正在生成 doc\constant-values.html...
正在构建所有软件包和类的索引...
正在生成 doc\overview-tree.html...
正在生成 doc\index-all.html...
正在生成 doc\deprecated-list.html...
正在构建所有类的索引...
正在生成 doc\allclasses-frame.html...
正在生成 doc\allclasses-noframe.html...
正在生成 doc\index.html...
正在生成 doc\help-doc.html...
正在生成 doc\stylesheet.css...
```

@Inherited 注释，此注释表示一个 Annotation 是否可以被继承下来。

```
package org.lxh.demo16.inheriteddemo ;
import java.lang.annotation.Retention ;
import java.lang.annotation.RetentionPolicy ;
import java.lang.annotation.Documented ;
import java.lang.annotation.Inherited ;
@Documented
@Inherited
@Retention(value=RetentionPolicy.RUNTIME)
public @interface MyInheritedAnnotation{
	public String name() ;

}
```

定义一个父类，在父类上使用此 Annotation/

```
package org.lxh.demo16.inheriteddemo ;
@MyInheritedAnnotation(name="李兴华")
public class Person{
};
```

定义子类

```
public class Student extends Person{
};
```

按照所解释的，使用 Inherited 声明的 Annotation 是可以被子类继承下来的。

```
import java.lang.annotation.Annotation ;
import org.lxh.demo16.inheriteddemo.MyInheritedAnnotation ;
public class ReflectInheritedDemo{
	public static void main(String args[]) throws Exception{
		Class<?> c = null ;
		c = Class.forName("org.lxh.demo16.inheriteddemo.Student") ;
		Annotation ann[] = c.getAnnotations() ; // 取得全部的 Annotation
		for(Annotation a:ann){ // 输出
			System.out.println(a) ;
		}
		// 继续取得此 Annotation 设置的内容
		if(c.isAnnotationPresent(MyInheritedAnnotation.class)){
			MyInheritedAnnotation mda = null ;
			mda = c.getAnnotation(MyInheritedAnnotation.class) ;
			String name = mda.name() ; // 取出 name 的内容
			System.out.println("name = " + name) ;
		}
	}
}
```

4、总结

1、 熟悉 Documented 注释的作用：加入说明信息
2、 熟悉 Target 的作用，并使用 Target 注释指定注释的使用位置
3、 如果一个 Annotation 要想被子类继承下来则使用 Inherited 注释说明
4、 反射机制对于操作 Annotation 是最重要的

5、预习任务

下一章内容

Java 数据库编程

E-MAIL：mldnqa@163.com www.MLDNJAVA.cn