1、课程名称：自定义 Annotation


JAVA培训先锋

JAVA 应用开发详解

第16章：Annotation
—— 自定义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

- 掌握自定义Annotation的简单格式
- 可以使用Annotation接收设置的内容
- 可以为Annotation中的变量指定默认值
- 可以使用枚举指定Annotation的取值范围
- 掌握Retention及RetentionPolicy的作用

3、具体内容


定义简单的Annotation

- Annotation定义格式：
  - [public] @interface Annotation名称{
    数据类型 变量名称() ;
    }

定义一个简单的 Annoatation

```
public @interface MyDefaultAnnotationNoneParam{

}
```

之后就可以直接在程序中使用"@Annotation 名称"的格式。

```
@MyDefaultAnnotationNoneParam
class Demo{
};
```

此时就表示在 Demo 类上使用 Annotation。

还可以向 Annotation 中设置设置，使用变量接收参数

```
public @interface MyDefaultAnnotationSingleParam{
        public String value() ;  // 接收设置的内容
}
```

在使用的时候就必须清楚的指名，变量的内容

```
@MyDefaultAnnotationSingleParam("李兴华")
class Demo{
};
```

或者也可以使用明确的标记，表示内容赋给哪个参数

```
@MyDefaultAnnotationSingleParam(value="李兴华")
class Demo{
};
```

以上的参数是要赋给 value 属性的。

既然可以设置一个参数，则也就可以同时设置多个参数。

```
public @interface MyDefaultAnnotationMoreParam{
        public String key() ;
        public String value() ;  // 接收设置的内容
}
```

此 Annnotation 在使用的时候需要设置两个参数，一个是 key，另外一个是 value。

```
@MyDefaultAnnotationMoreParam(key="MLDN",value="李兴华")
class Demo{
};
```

也可以设置一个数组进去。

```
public @interface MyDefaultAnnotationArrayParam{
        public String[] value() ;       // 接收设置的内容
}
```

接收的内容本身是一个数组类型，要传递数组，类似与之前的 SuppressWarnings

```
@MyDefaultAnnotationArrayParam(value={"MLDN","李兴华"})
class Demo{
};
```

以上所定义的全部的 Annotation 中有一个特点，所有的参数内容需要在使用注释的时候设置上去，那么也可以为一个参数设置默认的内容，在声明的时候使用 default 即可。

```
public @interface MyDefaultAnnotationValue{
        public String key() default "MLDN" ;   // 指定好了默认值
        public String value() default "李兴华" ; // 指定好了默认值
}
```

当在去使用此 Annontation 的时候就可以不用设置内容。

```
@MyDefaultAnnotationValue
class Demo{
};
```

没有设置内容，编译也不会出现任何的错误。

在操作中，对于一个 Annotation 而言有时候会固定其取值范围，只能取固定的几个值，那么这个时候实际上就需要依靠枚举，

```
public enum MyName{       // 定义枚举类型
        MLDN,LXH,LI ;
}
```

以后的 Annotation 的取值，只能从这三个值中取走。

```
public @interface MyDefaultAnnotationEnum{
        public MyName name() default MyName.LXH ;     // 指定默认值
}
```

此时以上的 Annotation 已经设置好了一个枚举中的内容作为默认值，那么外部使用此 Annotattion 的时候也需要从枚举固定取值

```
@MyDefaultAnnotationEnum(name=MyName.LI) // 只能从固定的枚举中取内容
class Demo{
};
```


Retention和RetentionPolicy

- 在Annotation，可以使用Retention定义一个Annotation的保存范围，此Annotation的定义如下：

```
@Documented
@Retention(value=RUNTIME)
@Target(value=ANNOTATION_TYPE)
public @interface Retention{
    RetentionPolicy value() ;
}
```

在以上的Rentetion定义中存在了一个RententionPolicy的变量，此变量用于指定Annotation的保存范围，RetentionPolicy包含三种范围


RetentionPolicy的三个范围

No.	范围	描述
1	SOURCE	此Annotation类型的信息只会保留在程序源文件之中（*.java），编译之后不会保存在编译好的类文件（*.class）之中
2	CLASS	此Annotation类型将保留在程序源文件（*.java）和编译之后的类文件（*.class）之中，在使用此类的时候，这些Annotation信息将不会被加载到虚拟机（JVM）之中，如果一个Annotation声明时没有指定范围，则默认是此范围
3	RUNTIME	此Annotation类型的信息保留在源文件（*.java）、类文件（*.class）在执行时也会加载到JVM之中

在三个范围中，最需要的关心的就是 RUNTIME 范围，因为在执行的时候起作用。

使用 Rentention 指定一个 Annotation 的范围，范围为 RUNTIME 范围

```
import java.lang.annotation.Retention ;
import java.lang.annotation.RetentionPolicy ;
@Retention(value=RetentionPolicy.RUNTIME)     // 表示此 Annotation 在运行时有效
public @interface MyDefaultRententionAnnotation{
        public String name() default "李兴华" ;
}
```

此 Annontation 可以在运行时起作用，而且在以后要讲解反射与 Annotation 的操作中，也必须使用此种范围。


内建Annotation的RetentionPolicy

- 三个内建的Annotation的定义：
  - Override定义采用的是@Retention(value=SOURCE)，只能在源文件中出现；
  - Deprecated定义采用的是@Retention(value=RUNTIME)，可以在执行时出现；
  - SuppressWarnings定义采用的是@Retention(value=SOURCE)，只能在源文件中出现。

4、总结

1、 Annortation 定义格式，及变量的类型：枚举、数组、普通变量

2、 Rentetion、RententionPolicy 的作用。

5、预习任务


下一章内容

反射与Annotation