# 注解

## 1.1 Retention说明

**@Retention(RetentionPolicy.RUNTIME)**

*/\*\*  
 \* Annotations are to be discarded by the compiler.  
 \*/****SOURCE***,  
  
*/\*\*  
 \* Annotations are to be recorded in the class file by the compiler  
 \* but need not be retained by the VM at run time. This is the default  
 \* behavior.  
 \*/****CLASS***,  
  
*/\*\*  
 \* Annotations are to be recorded in the class file by the compiler and  
 \* retained by the VM at run time, so they may be read reflectively.  
 \*  
 \** ***@see*** *java.lang.reflect.AnnotatedElement  
 \*/****RUNTIME***

@Retention(RetentionPolicy.SOURCE)   //注解仅存在于源码中，在class字节码文件中不包含

@Retention(RetentionPolicy.CLASS)     // 默认的保留策略，注解会在class字节码文件中存在，但运行时无法获得，

@Retention(RetentionPolicy.RUNTIME)  // 注解会在class字节码文件中存在，在运行时可以通过反射获取到

## 1.2 ElementType的用法

|  |  |
| --- | --- |
| 取值 | 注解使用范围 |
| METHOD | 可用于方法上 |
| TYPE | 可用于类或者接口上 |
| ANNOTATION\_TYPE | 可用于注解类型上（被@interface修饰的类型） |
| CONSTRUCTOR | 可用于构造方法上 |
| FIELD | 可用于域上 |
| LOCAL\_VARIABLE | 可用于局部变量上 |
| PACKAGE | 用于记录java文件的package信息 |
| PARAMETER | 可用于参数上 |

这里重点说明下：ElementType. PACKAGE。它并不是使用在一般的类中，而是用在固定的文件package-info.java中。这里需要强调命名一定是“package-info”。由于package-info.java并不是一个合法的类，使用eclipse创建类的方式会提示不合法，所以需要以创建文件的方式来创建package-info.java。

# Support Annotation

## 2.1 Nullness Annotations @NonNull

## 2.2 取值范围的注解

* FloatRange：指出一个被注解的元素应该是一个给定范围内的float值或double值。比如：

@FloatRange(from=0.0,to=1.0)

public float getAlpha() {

...

}

* IntRange：指出一个被注解的元素应该是一个给定范围内的int值或long值。
* IntDef：指出一个int类型的元素，它表示的是一个逻辑上的类型，并且它的值必须是被明确声明的常量之一。官方常使用这种方式使int类型代替enum类型。

使用实例：

//先定义 常量

public static final int SUNDAY = 0;

public static final int MONDAY = 1;

public static final int TUESDAY = 2;

public static final int WEDNESDAY = 3;

public static final int THURSDAY = 4;

public static final int FRIDAY = 5;

public static final int SATURDAY = 6;

//用 @IntDef "包住" 常量；

// @Retention 定义策略

// 声明构造器

@IntDef({SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY})

@Retention(RetentionPolicy.SOURCE)

public @interface WeekDays {}

@WeekDays int currentDay = SUNDAY;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

setCurrentDay(WEDNESDAY);

//声明变量

@WeekDays int today = getCurrentDay();

switch (today){

case SUNDAY:

break;

case MONDAY:

break;

case TUESDAY:

break;

case WEDNESDAY:

break;

case THURSDAY:

break;

case FRIDAY:

break;

case SATURDAY:

break;

default:

break;

}

}

public void setCurrentDay(@WeekDays int currentDay) {

this.currentDay = currentDay;

}

@WeekDays

public int getCurrentDay() {

return currentDay;

}

}

* StringDef：指出一个String类型的元素，它表示的是一个逻辑上的类型，并且它的值必须是被明确声明的常量之一。

## 资源注解

1. AnimatorRes ：指出一个integer的参数，成员变量，或方法返回值是一个animator资源的引用。
2. AnimRes：指出一个integer的参数，成员变量，或方法返回值是一个anim资源的引用。
3. AnyRes：指出一个integer的参数，成员变量，或方法返回值是一个任意资源类型的引用。
4. ArrayRes：指出一个integer的参数，成员变量，或方法返回值是一个array资源类型的引用。
5. AttrRes：指出一个integer的参数，成员变量，或方法返回值是一个attr资源的引用。
6. BoolRes：指出一个integer的参数，成员变量，或方法返回值是一个boolean资源的引用。
7. ColorRes：指出一个integer的参数，成员变量，或方法返回值是一个color资源的引用。
8. DimenRes：指出一个integer的参数，成员变量，或方法返回值是一个dimen资源的引用。
9. DrawableRes：指出一个integer的参数，成员变量，或方法返回值是一个drawable资源的引用（包括@mipmap）。
10. FractionRes：指出一个integer的参数，成员变量，或方法返回值是一个fraction资源的引用。
11. IdRes：指出一个integer的参数，成员变量，或方法返回值是一个id资源的引用。
12. IntegerRes：指出一个integer的参数，成员变量，或方法返回值是一个integer资源的引用。
13. InterpolatorRes：指出一个integer的参数，成员变量，或方法返回值是一个interpolator资源的引用。
14. LayoutRes：指出一个integer的参数，成员变量，或方法返回值是一个layout资源的引用。
15. MenuRes：指出一个integer的参数，成员变量，或方法返回值是一个menu资源的引用。
16. PluralsRes：指出一个integer的参数，成员变量，或方法返回值是一个plurals资源的引用。
17. RawRes：指出一个integer的参数，成员变量，或方法返回值是一个raw资源的引用。
18. StringRes：指出一个integer的参数，成员变量，或方法返回值是一个string资源的引用。
19. StyleableRes：指出一个integer的参数，成员变量，或方法返回值是一个styleable资源的引用。
20. StyleRes：指出一个integer的参数，成员变量，或方法返回值是一个style资源的引用。
21. TransitionRes：指出一个integer的参数，成员变量，或方法返回值是一个transition资源的引用。
22. XmlRes：指出一个integer的参数，成员变量，或方法返回值是一个xml资源的引用。

## 2.4 颜色注解

ColorInt

*public abstract void setTextColor(@ColorInt int color);*

## 2.5 线程注解

* BinderThread：指出被注解的方法应该只在binder线程中被调用。
* MainThread：指出被注解的方法应该只在主线程中被调用。
* WorkerThread：指出被注解的方法应该只在工作线程中被调用。
* UiThread：指出被注解的方法应该只在UI线程中被调用。

## 2.6 @Size 注解

*/\*\*  
 \* Denotes that the annotated element should have a given size or length.  
 \* Note that "-1" means "unset". Typically used with a parameter or  
 \* return value of type array or collection.  
 \*   
 \* Example:**\* public void getLocationInWindow(@Size(2) int[] location) {  
 \* ...  
 \* }  
 \* }*

## 2.7 @CallSuper注解

@CallSuper注解主要是用来强调在覆盖父类方法时，需要实现父类的方法，及时调用对应的super.\*\*\*方法,当用@CallSuper修饰了该方法，如果子类覆盖的后没有实现对呀的super方法会抛出异常。

## 2.8 @RequiresPermission

如果你的方法的调用需要调用者有特定的权限，你可以使用@RequiresPermission注解：

@RequiresPermission(Manifest.permission.SET\_WALLPAPER)

public abstract void setWallpaper(Bitmap bitmap) throws IOException;

如果你至少需要权限集合中的一个，你可以使用anyOf属性：

@RequiresPermission(anyOf = {

Manifest.permission.ACCESS\_COARSE\_LOCATION,

Manifest.permission.ACCESS\_FINE\_LOCATION})

public abstract Location getLastKnownLocation(String provider);

如果你同时需要多个权限，你可以用allOf属性：

@RequiresPermission(allOf = {

Manifest.permission.READ\_HISTORY\_BOOKMARKS,

Manifest.permission.WRITE\_HISTORY\_BOOKMARKS})

public static final void updateVisitedHistory(ContentResolver cr, String url, boolean real) {

对于intents的权限，可以直接在定义的intent常量字符串字段上标注权限需求(他们通常都已经被@SdkConstant注解标注过了):

@RequiresPermission(android.Manifest.permission.BLUETOOTH)

public static final String ACTION\_REQUEST\_DISCOVERABLE =

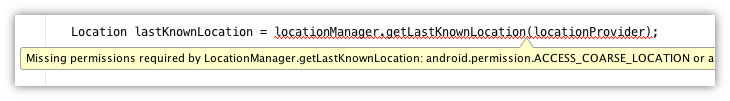
"android.bluetooth.adapter.action.REQUEST\_DISCOVERABLE";

对于content providers的权限，你可能需要单独的标注读和写的权限访问，所以可以用@Read或者@Write标注每一个权限需求：

@RequiresPermission.Read(@RequiresPermission(READ\_HISTORY\_BOOKMARKS))

@RequiresPermission.Write(@RequiresPermission(WRITE\_HISTORY\_BOOKMARKS))

public static final Uri BOOKMARKS\_URI = Uri.parse("content://browser/bookmarks");

[](http://www.flysnow.org/uploads/2015/08/image01.png)