极客学院
jikexueyuan.com

Java注解技术详解

Java注解技术详解 — **课程概要**

- Java 注解技术基本概念
- Java 标准注解
- Java 元注解
- Java 自定义注解
- Java 注解元素默认值

Java 注解技术基本概念

Java 注解技术基本概念

Annotation是Java5开始引入的新特征。中文名称一般叫注解。它提供了一种安全的类似注释的机制，用来将任何的信息或元数据（metadata）与程序元素（类、方法、成员变量等）进行关联：

- 概念
- 原理
- 应用场合

Java 标准注解

Java 标准注解

从Java5 版本开始，自带了三种标准注解类型：

标准注解：

- Override
- Deprecated
- SuppressWarinings

SuppressWarinings 示例：

```
@SuppressWarnings(value={"unchecked","fallthrough"})
public void test()  { /* method body  */ }
@SuppressWarnings({"unchecked","fallthrough"})
public void test()  { /* method body  */ }
@SuppressWarnings("unchecked")
public void test()  { /* method body  */ }
```

Java 元注解

Java 元注解

元注解的作用就是负责注解其他注解。Java5.0定义了4个标准的meta-annotation类型，它们被用来提供对其它 annotation类型作说明，Java5.0定义的元注解有以下四种类型：

- @Target
- @Retention
- @Documented
- @Inherited

Java 元注解 – **元注解之@Target**

@Target主要作用是用于描述注解的使用范围，即被描述的注解可以用在什么地方：

@Target使用范围：

- CONSTRUCTOR
- FIELD
- LOCAL_VARIABLE
- METHOD
- PACKAGE
- PARAMETER
- TYPE

@Target示例：

```
@Target(ElementType.TYPE)
public @interface Table {
   // 数据表名称注解，默认值为类名称
   public String tableName() default "className";
}

@Target(ElementType.FIELD)
public @interface NoDBColumn {    }
```

Java 元注解 – **元注解之@Retention**

@Retention主要表示需要在什么级别保存该注释信息，用于描述注解的生命周期：

@Retention取值：

- SOURCE
- CLASS
- RUNTIME

@Retention示例：

```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
public @interface Column {
   public String name() default "fieldName";
   public String setFuncName() default "setField";
   public String getFuncName() default "getField";
   public boolean defaultDBValue() default false;
}
```

Java 元注解 – 元注解之@Documented

@Documented用于描述其它类型的annotation应该被作为被标注的程序成员的公共API，因此可以被例如javadoc此类的工具文档化：

@Target示例：

```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface Column {
   public String name() default "fieldName";
   public String setFuncName() default "setField";
   public String getFuncName() default "getField";
   public boolean defaultDBValue() default false;
}
```

Java 元注解 – **元注解之@Inherited**

@Inherited 元注解是一个标记注解，@Inherited阐述了某个被标注的类型是被继承的。如果一个使用了@Inherited修饰的annotation类型被用于一个class，则这个annotation将被用于该class的子类：

@Inherited示例：

```
@Inherited
public @interface Greeting {
    public enum FontColor{ BULE,RED,GREEN};
    String name();
    FontColor fontColor() default FontColor.GREEN;
}
```

Java 自定义注解

Java 自定义注解

自定义注解主要分为以下两步：

- 通过@interface关键字声明注解名称、注解成员属性等
- 使用Java内置四个元注解对自定义标注的功能和范围进行约束

自定义注解的格式： public @interface 注解名 { 定义体 }

注解参数支持的数据类型：

- 所有基本数据类型
- String 类型
- Class 类型
- enum 类型
- Annotation 类型
- 以上所有类型的数组

自定义注解定义：

```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface FruitName {
    String value() default "";
}
```

自定义注解使用：

```
public class Apple {
    @FruitName("Apple")
    private String appleName;
    // ……
}
```

Java 注解元素默认值

Java 注解元素默认值

注解元素的默认值：注解元素必须有确定的值，要么在定义注解的默认值中指定，要么在使用注解时指定，非基本类型的注解元素的值不可为null：

```
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface FruitProvider {
    //供应商编号
    public int id() default -1;
    //供应商名称
    public String name() default "";
    //供应商地址
    public String address() default "";
}
```

Java注解技术详解

在本套课程中我们深入和详细的学习了Java的注解技术。你应当掌握了以下知识：

- Java 注解的基本概念
- Java 内置的标准注解
- Java 元注解
- Java 自定义注解

通过本课程的学习，你应该对Java的注解技术有了深入的了解以及如何通过使用JDK5的注解功能使开发更容易。如果你想继续提高，可以继续在极客学院学习Spring的其他相关课程。

极客学院

jikexueyuan.com

中国最大的IT职业在线教育平台