**Java运行时改变注解值**

2017年07月26日 00:17:13

阅读数：1825

可能有人会问为什么要在运行是改变注解值？有什么应用场景？

**场景：**

士兵表：   
   
士兵日志表：   
   
这里我用的是mysql，在公司开发用的是mysql，对每个主键都使用了sequence。从上面的字段可以看出，士兵日志表就是一个**id字段＋veteran对象**，定义日志这个bean时如下：

Veteran{

@ColumnAnnotation(fieldName="id", sequence="ve\_xxx")

private Long id

@ColumnAnnotation(fieldName="name");

private String name;

}

VeteranLog{

@ColumnAnnotation(fieldName="id", sequence="lg\_xxx")

private Long id;

@Compound

private Veteran veteran;

}

* 1

**注意：上面的注解全是自定义的，不用在意。**在保存一个士兵日志对象时我需要解析veteran对象中的所有属性值，并且要清除它里面的id属性注解的sequence属性值为空。

**实践环节：**

下面是从stackoverflow三找到的一个简单demo并做了一定修改：   
先定义四个注解：

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.TYPE)

public @interface ClassAnnotation {

String value() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.FIELD)

public @interface FieldAnnotation {

String value() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.METHOD)

public @interface MethodAnnotation {

String value() default "";

}

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.FIELD)

public @interface CompoundAnnotation {

String value() default "";

}

* 1

AnnotationTestClass类（包含复合对象，要进行递归解析的对象）

@ClassAnnotation("test class")

public class AnnotationTestClass {

@FieldAnnotation("test field")

public Object obj;

//实例化该对象是为了防止运行时通过反射拿到该对象得到的是一个空值，导致拿到该对象属性注解值抛出NullerPointException

@CompoundAnnotation

public SecondAnnotationTestClass secondTestClass = new SecondAnnotationTestClass();

@MethodAnnotation("test method")

public void testMethod(){

System.out.println("this is the test method invocation");

}

}

* 1

SecondAnnotationTestClass类（要在运行期通过反射获取到该定义的实例，通过实例获取到属性注解，动态改变注解值）

public class SecondAnnotationTestClass {

//该注解的值将通过实例在运行期通过反射获取并被改变

@FieldAnnotation("second test field")

public Object field;

}

* 1

MainClass类（核心测试类）

public class MainClass {

public static Object changeAnnotationValue(Annotation annotation, String key, Object newValue) throws Exception{

InvocationHandler handler = Proxy.getInvocationHandler(annotation);

Field f;

try {

f = handler.getClass().getDeclaredField("memberValues");

} catch (Exception e) {

e.printStackTrace();

throw new IllegalStateException(e);

}

f.setAccessible(true);

Map<String, Object> memberValues;

memberValues = (Map<String, Object>) f.get(handler);

Object oldValue = memberValues.get(key);

if(oldValue == null || oldValue.getClass() != newValue.getClass()){

throw new IllegalArgumentException();

}

memberValues.put(key, newValue);

return oldValue;

}

public static void main(String[] args) throws Exception {

AnnotationTestClass annotationTestClass = new AnnotationTestClass();

ClassAnnotation classAnnotation = AnnotationTestClass.class.getAnnotation(ClassAnnotation.class);

System.out.println("class annotation old value : " + classAnnotation.value());

changeAnnotationValue(classAnnotation, "value", "class annotation new value");

System.out.println("modified annotation new vlaue : " + classAnnotation.value());

System.out.println("============");

Field field = AnnotationTestClass.class.getField("obj");

FieldAnnotation fieldAnnotation = field.getAnnotation(FieldAnnotation.class);

System.out.println("field annotation old value : " + fieldAnnotation.value());

changeAnnotationValue(fieldAnnotation, "value", "field annotation new value");

System.out.println("modified annotation new vlaue : " + fieldAnnotation.value());

Field[] declaredFields = annotationTestClass.getClass().getDeclaredFields();

for(Field f : declaredFields){

f.setAccessible(true);

CompoundAnnotation fa = f.getAnnotation(CompoundAnnotation.class);

if(null != fa){

//反射拿到这个复合对象

Object compoundAnnotatedObject = f.get(annotationTestClass);

Field onlyField = compoundAnnotatedObject.getClass().getField("field"); //复合对象中唯一的一个属性

FieldAnnotation comFieldAnnotation = onlyField.getAnnotation(FieldAnnotation.class);

System.out.println("compound object field with annotation value is : " + comFieldAnnotation.value());

changeAnnotationValue(comFieldAnnotation, "value", "new second test field");

System.out.println("after changed compound object field with annotation value is : " + comFieldAnnotation.value());

}

}

System.out.println("============");

Method method = AnnotationTestClass.class.getMethod("testMethod");

MethodAnnotation methodAnnotation = method.getAnnotation(MethodAnnotation.class);

System.out.println("method annotation old value : " + methodAnnotation.value());

changeAnnotationValue(methodAnnotation, "value", "method annotation new value");

System.out.println("modified annotation new vlaue : " + methodAnnotation.value());

}

}

* 1

问题：上面的handler.getClass().getDeclaredField(“memberValues”);中membersValue是个什么东西？   
membersValue是java注解的私有map，那里面存放着注解的键值对，任何注解的属性定义及其赋值最终都会以键值对的形式存放到这个map中，通过上面的方法可以通过java反射的方式获取到某个注解中的所有键值对并替换某个key的value。

**运行结果：**